PROJECTE FINAL DE CARRERA

MIDI Loop-Station

Estudis: Enginyeria de Telecomunicació

Autor: Oscar Antolí

Director: Sergi Bermejo

Any: 2010/2011

Índex general

Col·laboracions

Resum del Projecte

Aquest projecte final de carrera va partir d'una idea molt ambiciosa: el desenvolupament d'un sistema capaç de (1) enregistrar i processar una senyal d'àudio provinent d'un clarinet i transformar-la en una senyal MIDI que (2) fós enviada a un robot que, mitjançant un sistema mecànic, manipulés un clarinet baix i reproduís la melodia inicial. El resultat seria la creació d'un robot-músic que emulés el comportament dels Loop-Station existents al mercat.

Un aparell Loop-Station típic repeteix una seqüència prèviament interpretada, de tal manera que una persona es pot gravar a si mateix i després tocar damunt del que s'ha enregistrat. Aquests sistemes capten la senyal d'àudio durant un cert temps, que ve determinat normalment per un sistema de pedals, i després la reprodueixen en forma de bucle.

Aquest projecte es centra en el mòdul de processament de senyal que permet la transformació àudio-MIDI de la seqüència original -interpretada per un clarinet o instrument similar-. Partint d'una seqüència musical curta interpretada per l'usuari, el sistema la enregistra, processa i finalment envia una senyal de sortida que es reprodueix en forma de bucle. La senyal de sortida serà una sèrie d'instruccions MIDI *(Musical Instrument Digital Interface*) que podran ser reproduïdes per qualsevol dels múltiples instruments virtuals que existeixen avui en dia -del clave a la guitarra elèctrica passant pel dijeridoo-.

Des d'un punt de vista tècnic, l'algoritme s'ha desenvolupat sobre el programari Matlab. Aquest permet que la senyal inicialment captada a través d'un micròfon connectat a la targeta de so, sigui processada per l'algoritme. L'algoritme reconeix les notes que componen la seqüència original i n'extreu les característiques musicals bàsiques: nota, volum i vibrat. Finalment, utilitzant complements Matlab desenvolupats al marge d'aquest projecte, s'envien les senyals MIDI de sortida conservant les característiques musicals de la senyal original.

Per últim, es descriu la línia a seguir per tancar el cercle i aconseguir l'ambiciós objectiu inicial de crear un robot-músic que reproduís la senyal original en un clarinet baix. Es proposa integrar en un DSP (digital signal processor) el mòdul de processament abans descrit. Aquest esdevindria el cervell que enviaria les instruccions MIDI a un segon controlador responsable del sistema mecànic que manipularia l'instrument. A part d'això, també es proposa la implementació d'un algoritme de separació cega de fonts, per tal d'obtenir les seqüències originals quan dues o més fonts són enregistrades

simultàniament per diversos micròfons. La seqüència original a extreure estarà bàsicament contaminada pel so de l'altra font, pels rebots amb les parets i per l'eco i les característiques de la sala.

Abstract

This master thesis was triggered by a very ambitious idea: the creation of a system capable of (1) recording and processing an input audio signal generated by a clarinet, transform it into a MIDI output signal and (2) eventually send it to a robot that plays the initial melody on a bass clarinet. Thereby the aimed result is a "musician-robot" that behaves similarly to the loop-stations available in the market.

Typical loop-stations allow musicians to record a sequence and repeat it in loop-mode. The recorded sequence is then used as a base while the musician has total freedom to play on top of it. In most common loop-stations, the recording time is determined by a pedal that is controlled by the musician.

This thesis focuses on the signal-processing module that transforms the original audio input into a MIDI output signal. First of all, the system records and processes a short melody played by the user on a clarinet or similar instrument. Then the input signal is transformed and reproduced in loop-mode. The output is a series of MIDI instructions that can be reproduced by any virtual instrument (e.g. electric guitar, didgeridoo or harpsichord).

From a technical standpoint, the signal processing algorithm has been developed over Matlab. Sequentially, the audio signal is captured by a microphone plugged into the computer's sound card. Then it is processed by the abovementioned algorithm that breaks down the melody into individual notes and extracts its basic musical properties: pitch, volume and vibration. Eventually, using complementary modules developed aside from this project, the signal is sent out keeping all the original musical properties.

Eventually, the thesis outlines the way forward to complete the original ambitious idea of creating a "musician-robot" that would play the original melody on a bass-clarinet. It is suggested to integrate the signal processing module in a DSP. It would then become the brain of the robot that would send the MIDI signals to a second controller responsible for the mechanical system that would manipulate the bass clarinet. The mechanical system is also briefly described. It also proposes the implementation of a “Nonlinear Blind Source Separation” algorithm, in order to obtain an original sequence when two or more sources are simultaneously recorded by several microphones. The original sequence will be basically contaminated by the sound of the other source, by the bounce of the walls and by the echo and the characteristics of the room.

1. Introducció

1.1 Context del projecte

Aquest projecte neix bàsicament de les pròpies inquietuds musicals i tècniques; es tracta d'un projecte personal. Com estudiant de Telecomunicació a l'Escola Superior de Telecomunicació de Barcelona (ETSETB) i de clarinet a l'Escola Superior de Música de Catalunya (ESMUC), la idea de poder fusionar aquestes dues disciplines, en la direcció de poder improvisar motius curts per ésser reproduïts amb altres instruments controlats per autòmats i poder tocar damunt, resultava una idea prou estimulant amb la qual començar a treballar.

Inicialment, el projecte es va recolzar en nombrosos articles publicats per la Universitat Waseda de Tokyo que ja fa anys que treballa amb robots musicals i amb interaccions entre ells. Més concretament van dissenyar un robot que interpreta una peça prèviament escrita, s'anomena WAS-I (Petersen, 2009). En el capítol 2, es veuen els mecanismes emprats en la realització del sistema d' embocadura.

Animat amb aquesta possibilitat, s'inicia per una banda el treball en l'algoritme de detecció de notes i per l'altra, s'intenta crear un prototip amb prou menys mitjans que la universitat abans esmentada.

1.2 Objectius

L'objectiu principal d'aquest projecte passa per la codificació de les senyals d'àudio al format MIDI. Per aconseguir això es proposa un codi desenvolupat amb el programari Matlab on s'analitzarà la senyal que es capta amb un micròfon. La meta que es pretén aconseguir és descriure característiques musicals com són: l'afinació, el volum i el tempo. Tanmateix, es pretén elaborar una interfície gràfica que pugui controlar tots els elements (funcions, transferències de dades, temporitzadors...) perquè permeti a l'usuari interactuar fàcilment amb les opcions que el programa proporciona.

El segon objectiu serà, un cop obtingut la caracterització de la senyal, crear la seqüència MIDI perquè pugui ser enviada a qualsevol agent extern per poder-la reproduir. Aquestes senyals poden ser reconegudes per molts sistemes, com per exemple sintetitzadors, programes d'edició d'àudio, etc.

El tercer objectiu serà la implementació d'un algoritme per separar dues fonts sonores que emeten a l'hora. El propòsit serà la obtenció d'una de les dues senyals originals descontaminada de l'altra, de tal manera es pugui analitzar sense problemes.

Finalment, com a últim objectiu, es pretén elaborar un petit prototip que permeti controlar el clarinet baix. Aconseguir la coordinació necessària entre els elements utilitzats perquè es pugui interpretar la seqüència enviada des del controlador.

1.3 Estructura de la memòria

La part central de la memòria d'aquest projecte es distribueix en cinc blocs. En el primer bloc –Capítol 2- s'explicaran les fonts, el context i els materials que s'han utilitzat per l'elaboració d'aquest projecte. A part també es mencionaran, quines idees d'articles s'han tingut en compte per programar les funcions que permetran captar, processar i enviar les senyals.

En el segon bloc s'explica el codi elaborat amb el programari Matlab –Capítol 3-. Es tracta de veure les funcions utilitzades per tal d'extreure les característiques que es necessiten de la senyal. En aquest apartat es treballa amb llenguatge Matlab.

Pel que fa al tercer bloc –Capítol 4-, s'explica la interfície gràfica que ha estat implementada per a poder interactuar amb tots elements que apareixen en aquest projecte i com es transmeten les senyals. També es veurà com aquest transforma les dades rebudes en instruccions MIDI perquè puguin ésser reconegudes per qualsevol dispositiu compatible.

El capítol 5 proposa el sistema a seguir per tal d' aconseguir el Loop-Station real amb el clarinet baix com a instrument controlat automàticament. S'expliquen els passos que s'han seguit per obtenir aquest robot músic.

Per últim –capítol 6-, s'exposaran les conclusions i futures línies de treball a seguir per tal d'aconseguir desenvolupar el projecte amb la seua totalitat.

2. Idees bàsiques

2.1 Conceptes musicals bàsics

En aquest apartat s'explicaran els conceptes musicals bàsics per entendre el desenvolupament del projecte.

- Volum: és la qualitat acústica que permet diferenciar entre so o sorolls forts o fluixos. Aquest paràmetre pot variar amb la pressió del so, la freqüència fonamental, l'ample de banda i la durada.

- *Crescendo*: terme italià utilitzat per indicar un canvi de volum de fluix a fort.

- *Pitch* o altura musical: és la percepció auditiva subjectiva que permet ordenar els sons dins d'una determina escala. Es pot quantificar com una freqüència, però no és una propietat física purament objectiva, fa referència a l'aspecte psicoacústic de so.

- Vibrat: són petites oscil·lacions del *Pitch*, petites desafinacions, que s'executa normalment per donar més expressivitat a la peça musical. Es caracteritza per la extensió (el rang de freqüència que abraça) i la velocitat d'execució.

- Durada: és la base del ritme, és l'interval de temps on una nota o un silenci està produint-se.

- Tempo: és el paràmetre que defineix la velocitat de la peça. És un dels elements que defineixen més el caràcter de l'obra. Així doncs una mateixa melodia interpretada més ràpid -a un tempo més alt- pot provocar una sensació totalment diferent a un tempo lent.

- Timbre: és la propietat del so que permet distingir entre diverses fonts sonores ja siguin de vent, de corda de percussió... Els factors que determinen aquest timbre són bàsicament característiques espectrals i d'envolupant de l'ona sonora.

- *Bend* o *glissando*: és la manera com varia la freqüència entre notes. Com es varia la freqüència per passar d'una nota a una altra o dins d'una mateixa nota –en aquest últim cas seria el vibrat-.

- Mordent: és una nota de durada molt curta que adorna la nota que segueix. Normalment el *pitch* d'aquestes dues és molt proper.

2.2 Concepte del robot

Durant les últimes dècades es porta fent un treball d'investigació prou ampli pel que fa al tema de robots musicals. Un exemple dintre d'aquest camp, i com a referència de partida d'aquest projecte, són els treballs publicats per un equip de recerca de la Universitat de Waseda al Japó. Un dels seus articles (Petersen, 2009) descriu la manera com s'ha realitzat el prototip d'un robot, anomenat WAS-1, que interpreta una seqüència musical amb el saxo alt. Anteriorment a aquest treball, però, ja s'havien implementat robots per tocar el teclat, el violí i la flauta (Solis, 2007).

El robot WAS-1 està dissenyat per a reproduir l'anatomia i fisiologia dels òrgans humans que intervenen a l'hora de fer sonar l'instrument. Tal i com es pot veure en la figura 2-1, presenta un sistema que emula el comportament humà i posteriorment descriu detalladament com s'han implementats aquests òrgans artificials, cadascun dels quals presenta uns certs graus de llibertat (DOF).

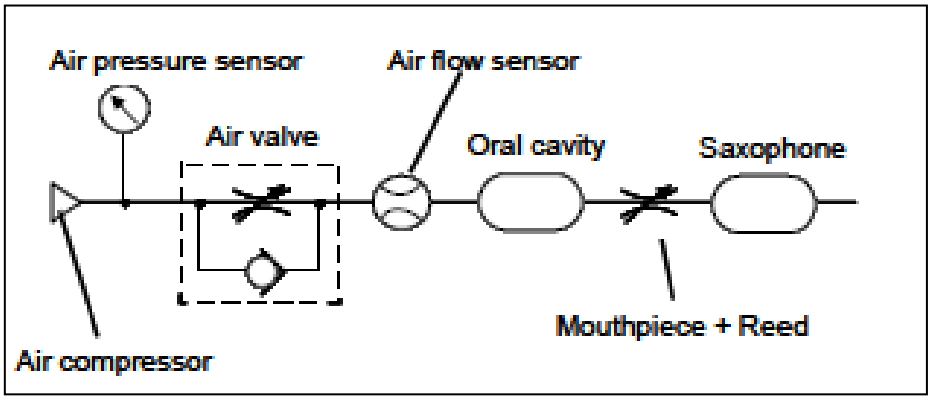


Figura 2-1: Esquema de les parts del robot WAS-1 (Petersen, 2009).

Segons l'esquema a dalt proposat per a implementar aquest sistema es precisen dels següents òrgans artificials: llavis (1 DOF), llengua (1 DOF), cavitat oral i pulmons (1 DOF), i vàlvula d'aire (1 DOF). En la figura 2.2, per exemple, es pot veure com ha estat implementat la part del llavi, amb goma termoplàstica i la barra de metall que controla la pressió exercida sobre el mateix llavi mitjançant un motor. A part de tot el sistema de producció del so també és necessari la implementació del sistema que permeti moure les tecles mitjançant un sistema de palanques amb un total de 11 graus de llibertat. Per últim, s'ha de determinar quina ha de ser la pressió dels llavis i la llengua respecte a la de l'aire perquè la nota soni correctament; en aquesta direcció, els dissenyadors del robot WAS-1 determinaren les fórmules per obtenir la pressió necessària perquè puguin sonar les notes correctament.

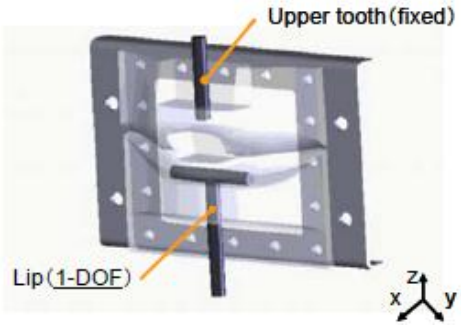


Figura 2-2: Esbós de la boca termoplàstica (Petersen, 2009).

Per al robot que toca la flauta s'ha anat un pas més enllà i s'investigà com es mou la glotis de la persona per a, posteriorment, crear un prototip de gola que permet obtenir més acuradament una interpretació més fidedigna i, així, possibilita l'estudi de l'obtenció de vibrats més reals. Per contra, el model WAS-1, de moment, només s'han quedat amb la boca.

Un altre aspecte molt interessant dels treballs realitzats per aquesta universitat japonesa que també s'ha tingut en compte és la possible interacció entre aquests robots músics. En un dels articles (Solis, 2006) es presenta un mecanisme perquè permeti la interactuació entre el robot saxofonista -WAS-1- i el robot flautista -WF-4RIV-. Aquest últim que està constituït per molts sensors de visió, de so i de moviment. En l'article es deixa la porta oberta a aquesta possibilitat però només s'experimenta amb duets.

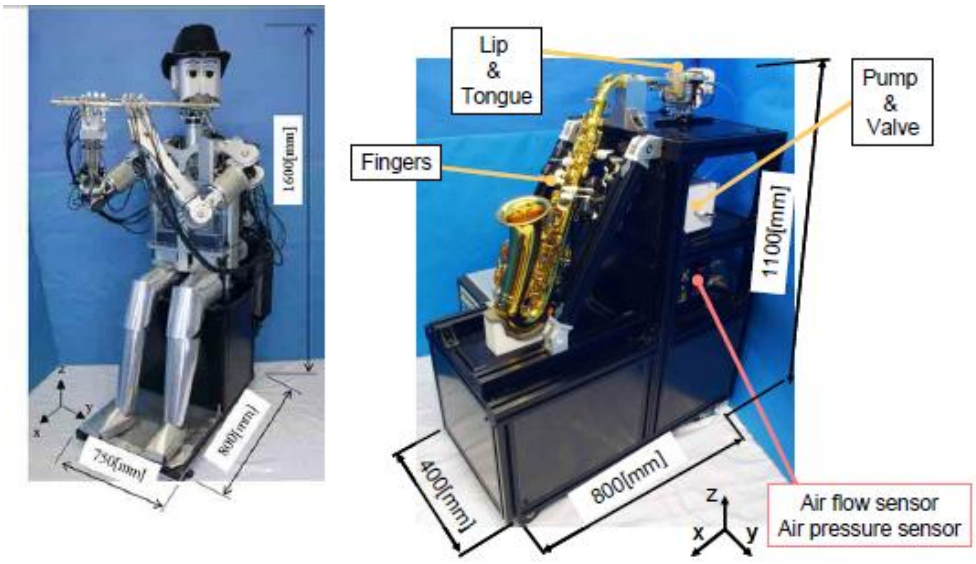


Figura 2-3: Robots WF-4RIV i WAS-1 (Solis, 2006).

2.3 Algoritme de processament

Una part molt important d'aquest projecte passarà per esbrinar la freqüència fonamental de les notes que es captaran i, tot seguit, com es poden codificar els efectes com el vibrat, el volum i la durada. En aquest apartat es presenten els algoritmes que han inspirat o s'han tingut en compte a l'hora d'elaborar el codi.

Una de les primeres referències alhora de plantejar com elaborar l'algoritme va ser un article on es proposa un sistema per transformar les notes àudio a MIDI (Arvin, 2009). En aquest treball es descriu tot el procés de captació amb el micròfon, filtratge i amplificació i posteriorment processat, que ho fa mitjançant un DSP (Processador de senyal digital).

L'algoritme es basa en crear una senyal digital de la seqüència que es capta del micròfon a partir d'un llindar determinat tal com es veu en la figura 2-4.

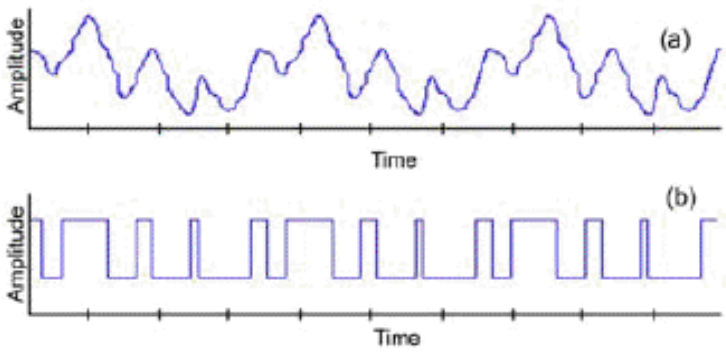


Figura 2-4: Senyal analògica captada (a) i transformada a impulsos digitals (b) (Arvin, 2009).

La secció del senyal s'extreu per l'amplitud d'un grup de pics significatius; tots aquells amb nivells absoluts que siguin majors que el valor especificat seran seleccionats. D'altra banda, els pics de cada interval que tinguin una freqüència fonamental similar es subdivideixen en una freqüència fonamental per a cada pic. Aquesta subdivisió permet trobar la diferència entre els pics principals dins l'interval que s'estudia. Així doncs, el resultat dependrà de la longitud de finestra que s'esculli.

En aquest treball es proposen dues maneres per determinar aquesta longitud de finestra de mostres enregistrades. En cadascuna d'aquestes finestres es calcularà la freqüència fonamental del so captat. Es diferenciaran finestres de mides estàtiques i dinàmiques:

- ***Mida estàtica de finestra:***

En aquest enfocament, les mostres capturades es divideixen en finestres de mida fixa. Els *timers* del microcontrolador es fan servir per controlar la duració de cada cicle. Una matriu de dues dimensions s'utilitza per a la gravació de mostres capturades. La durada es guarda com la primera dimensió i el temps d'inici de cada pols com a segon. Al final de cada finestra, es cridarà la funció de càlcul de la freqüència, que trobarà els increments de temps màxims que s'utilitzaran per al càlcul de la freqüència.

La mida de cada finestra és un paràmetre important per a l'estimació de la freqüència fonamental. Les durades de les notes interpretades és l'altre paràmetre. Quan la grandària de les finestres es fa més gran, es caputeren més mostres i la sortida serà més fidel a la freqüència real. En contrapartida, però, si la mida de la finestra és gran, el processador no tindrà temps suficient per processar les dades capturades. A més, la selecció d'una finestra de grans dimensions pot provocar la pèrdua de les notes amb durades curtes.

- ***Mida dinàmica de finestra:***

En aquest enfocament, la mida de la finestra no és constant. La seua aplicació és més complexa que l'enfocament de finestra estàtica. S'utilitzen diversos paràmetres per canviar la mida.

La primera raó per tancar una finestra és per la captura de polsos d'ample similars. Quan es capturen dos polsos de màxim igual en un període, la finestra es pot tancar i pot començar una nova finestra. En aquest enfocament, les notes de curta durada no s'ignoraran. Aquesta funció utilitza una variable comptador i només s'afegirà un procés per trobar el pols màxim.

La segona raó serà l'espai entre dues notes. Aquesta funció detecta les àrees de silenci i crearà una funció de *Time-out* que permetrà tancar una finestra si es reben només polsos que són més petits que les freqüències de les notes.

La Figura 2-5 mostra els resultats de les proves realitzades amb aquest mètode. En aquests experiments, els temps de durada de finestra estàtica són de 20, 100 i 500 ms. A cada algorisme, es van tocar 200 notes i es distingeixen entre notes de curta i de llarga durada per calcular en el percentatge de notes correctament transcrites. Per cada tipus es pot veure clarament com els resultats varien. La tècnica de captura de finestres dinàmic pot oferir resultats millors, tot i que es deixa per futures investigacions. Com es pot veure, tot i que aquest sistema presenta un algoritme molt senzill, la taxa de reconeixement és del 80% en el millor dels casos.

PERCENTAGE OF CORRECTLY TRANSCRIBED NOTES

Window Size	Short duration notes	Long duration notes
10 ms	30 %	40 %
100 ms	75 %	80 %
500 ms	60 %	70 %

Figura 2-5: Resultats de detecció del sistema (Arvin, 2009).

Nous algoritmes que aconsegueixen una millora del reconeixement els trobarem en alguns dels escrits publicats per la doctora Judith Brown. Com a primer mètode d'anàlisi s'estudià l'autocorrelació estreta –*"narrowed" autocorrelation*- (Brown, 1988), que és útil quan la duració és al menys diverses vegades el període a estudiar, fet que per a l'anàlisi del so es compleix amb escreix. Tot i això, es va trobar com a primer problema la identificació del pic, que pot ser errònia degut a l'aparició d'un soroll periòdic, inclús en els casos que presenten una amplitud petita.

La funció d'aquesta autocorrelació *estreta* es pot obtenir mitjançant la inclusió, a més del producte habitual $f(t)\cdot f(t+\tau)$, dels productes de la forma $f(t)\cdot f(t+2\tau)$, $f(t)\cdot f(t+3\tau)$, ... , $f(t)\cdot f(t+(N-1)\tau)$. Considerant la funció autocorrelació (2.1).

$$g(\tau)=\int_{-\infty}^{\infty} f(t)\cdot f(t-\tau)dt \qquad (2.1)$$

Per obtenir la aquesta funció es remarca que és millor calcular primerament la funció $<|f(t)+f(t-\tau)|^2>$, els productes creuats de la qual donaran l'autocorrelació *estreta*.

Si enlloc d'aquesta autocorrelació es considera aquesta nova funció (2.2).

$$S_N(\tau)=f(t)+f(t-\tau)+f(t-2\tau)+\cdots+f(t-(N-1)\tau) \qquad (2.2)$$

El quadrat d'aquesta nova funció (2.3) inclourà el producte de cada parell de termes. La *N* definirà el nombre de parelles que s'agafaran per definir aquesta autocorrelació *estreta*.

$$|S_N(\tau)|^2=<|f(t)+f(t-\tau)+f(t-2\tau)+\cdots+f(t-(N-1)\tau)|^2> \qquad (2.3)$$

Per a funcions periòdiques de període T, tots els termes estaran en fase per als retards τ iguals a un integral múltiple de T. Si considerem un petit canvi de τ per $(T+\Delta)$, el n-terme restarà $(n-1)\Delta$ fora de fase amb el primer terme. D'aquí que aquesta funció se'n va de fase molt ràpidament fet que fa que el pic sigui molt estret.

En (Brown, 1988) es fa un anàlisi de senyals mostrejats a 32 Khz, agafant sobretot sons de violí. La figura dreta següent correspon a gràfiques d'aquesta correlació modificada per a *N=2, N=5* i *N=10,* és a dir, utilitzant dos, cinc o deu termes. Es pot veure que realment el pic és molt més estret. La imatge de la dreta correspondria a un exemple de nota RE[5] vibrat amb el violí.

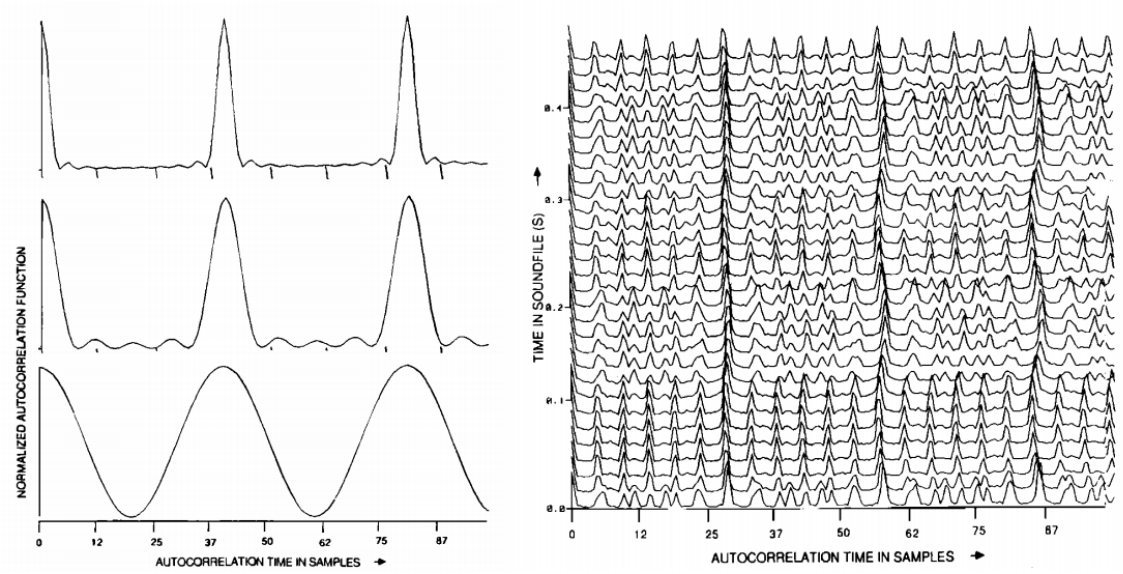


Figura 2-6. Funcions d'autocorrelació (Brown, 1988).

El segon dels algoritmes que s'aplicaran és basa en crear contenidors de diferents longitud dins de l'espectre de freqüències, és a dir que abans de calcular la transformada, es decideix a quina distància es separen els contenidors. Això es farà canviant a cada pas el nombre de mostres de la hipotètica transformada de Fourier, de tal manera que situem la mostra a la freqüència que es vol estudiar.

Tal com explica la doctora Brown (Beauchamp, 2007), la transformada discreta de Fourier no és la més adequada per a aplicacions musicals. Com se sap, l'escala musical que coneixem funciona exponencialment, és a dir, que quan volem fer una nota i la seua octava superior, la freqüència d'aquesta última es multiplica per 2. Llavors allò que passa és que a mesura que les notes es van fen més agudes tenim un interval cada vegada més petit i això fa que cada vegada hi hagin més mostres a analitzar en un to amb la DFT, cosa que la fa ineficient.

En (Beauchamp, 2007) s'estudia la implementació de la que s'anomena transformada constant Q. Així doncs, tenint en compte que la freqüència d'una nota de l'escala (f_k), aquesta es pot obtenir com a producte d'una freqüència de referència i un factor:

$f_k = (2^{\frac{1}{n}})^k \cdot f_{min}$. On n serien les divisions en què dividiríem l'octava, f_{min} seria la freqüència base –per exemple 440 per un A[4] – i k seria la distancia en semitons que separa la freqüència actual de la base. Considerant la distància entre dos semitons consecutius es pot obtenir com (2.4).

$$\Delta f_k = f_{k+1} - f_k = (2^{\frac{1}{n}})^{k+1} \cdot f_{min} - (2^{\frac{1}{n}})^k \cdot f_{min} = \left(2^{\frac{1}{n}} - 1\right) \cdot (2^{\frac{1}{n}})^k \cdot f_{min} = \left(2^{\frac{1}{n}} - 1\right) \cdot f_k \quad (2.4)$$

Llavors d'aquí surt $f_k/\Delta f_k = \left(2^{\frac{1}{n}} - 1\right)^{-1} = Q$. Aquest és un valor continu que serà la constant Q, que no és més que la ràtio de la freqüència respecte l'ample de banda. Δf_k, és la resolució o ample de banda que serà diferent entre contenidors de freqüència. Aquesta resolució dependrà, doncs, de la freqüència que s'estigui analitzant $(\Delta f_k = f_k/Q)$.

Una de les principals diferències entre la transformada Q i la transformada ràpida de Fourier (DFT) serà que en la DFT aquest ample de banda és constant. Aquest valor no és més que la freqüència de mostreig entre el número de mostres –amplada de finestra-. En canvi, aquest número de mostres en la transformada Q serà variable tal com mostra la fórmula (2.5)

$$N[k] = \frac{f_s}{\Delta f_k} = f_s/\left(\frac{f_k}{Q}\right) = f_s \cdot \frac{Q}{f_k} \quad \rightarrow \quad f_k = Q \cdot f_s/N[k] \quad (2.5)$$

La pulsació digital serà doncs: $2\pi \cdot Q \cdot f_s/N[k]$.

A partir de la DFT (2.6), es reescriurà de tal manera que compleixi els requisits de la nova transformada Q (2.7).

$$X[k] = \sum_{n=0}^{N-1} w[n]x[n]\, e^{-\frac{2j\pi kn}{N}} \quad (2.6)$$

$$X^Q[k] = \frac{1}{N[k]} \sum_{n=0}^{N[k]-1} w[n,k]x[n]\, e^{-2j\pi kn\ /N[k]} \quad (2.7)$$

Ja que el número de mostres canviarà per cada valor de k, serà molt important normalitzar aquesta transformada pel nombre variable de mostres amb les quals es calcularà. La taula 2-1 ens fa una comparativa de les característiques d'aquesta transformada respecta la DFT.

La funció w no és més que la funció d'enfinestrament per la senyal a analitzar, en aquest article, Brown utilitza la finestra de Hamming:

$$w[n,k] = \alpha + (1-\alpha)\cos\left(\frac{2\pi n}{N[k]}\right) \quad (2.8)$$

On $\alpha = 25/46$ i $0 \le n \le N[k] - 1$

Parameter	DFT	Constant-Q transform
Frequency f_k	$k\Delta f$ (Linear in k)	$(2^{1/24})^k \cdot f_{\min}$ (Exponential in k)
Window size	Constant $= N$	Variable $= N[k] = f_s Q / f_k$
Resolution Δf	Constant $= f_s / N$	Variable $= f_k / Q$
$\frac{f_k}{\Delta f_k}$	Variable $= k$	Constant $= Q$
Cycles in window	Variable $= k$	Constant $= Q$

Taula 2-1: Comparació entre la DFT i la transformada Q (Beauchamp, 2007).

En una altra part de l'article es tracta l'anàlisi més exhaustiu del so. En concret s'analitza el vibrat amb violoncel i flauta amb una resolució més alta. En aquest s'utilitzarien els cents, on dividiríem el semitò en 100 parts, quedant-se llavors amb una *n* igual a 1200: els 12 semitons de l'escala per una subdivisió de 100 parts. En altres treballs de Brown, com ara (Brown, 1990), es poden trobar més exemples d'anàlisis d'aquesta transformada Q.

Per altre banda, un article igualment rellevant per a aquest projecte n'és publicat a la Universitat de Waseda (Qi, 2002). Aquest proposa un sistema de so, amb bases de dades, que és capaç d'extreure les dades emmagatzemades utilitzant el so com a element clau per a la consulta. Presenta una sèrie de característiques mitjançant les quals es poden classificar els sons i estableix relacions entre el timbre, l'afinació, el volum i la durada subjectiva analitzant la corba de l'envolupant; també aclareix que hi ha una forta correlació entre la durada de l'atac i el *sustain* de l'ona d'un so amb el seu timbre. D'aquesta manera, quan s'extreuen les característiques de l'ona original, el so es simplifica com es mostra a la figura 2-7.

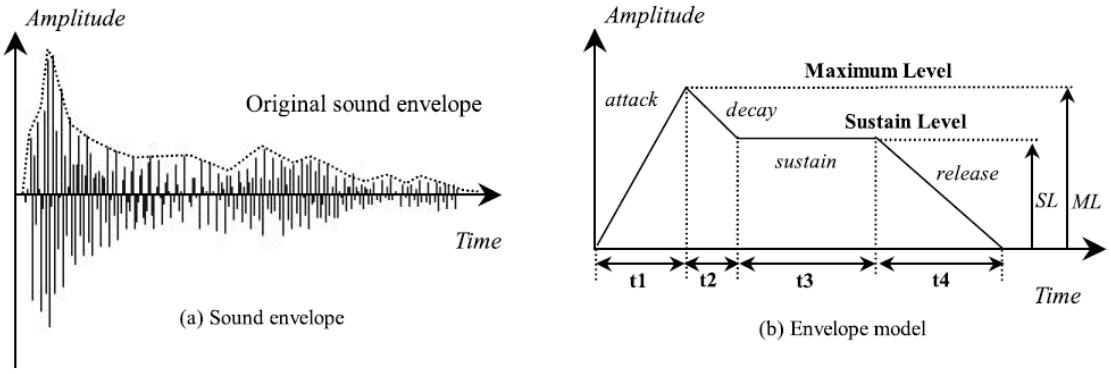


Figura 2-7: Envolupant del so (Qi, 2002). [Nota: les característiques extretes són la durada de l'atac (t1), el deteriorament durada (t2), la durada sostinguda (t3), la durada de l'alliberament (t4) i la relació entre el nivell sostingut (r1).]

De cada envolupant espectral es pot calcular un factor que es defineix com harmonicitat, que és la suma de l'energia de la freqüència fonamental i totes les freqüències harmòniques respecte a la suma de tota l'energia. Aquest paràmetre indica la proporció de components harmònics en el so i serà important per a generar la impressió d'*agradable*.

Tanmateix, tenint en compte aquests paràmetres, l'article descriu aquest sistema com a mitjà d' investigació bàsica amb l'objectiu de construir una base de dades que serveixi per a futures recerques. Són un exemple d'aquestes: l'avaluació del sistema proposat en una base de dades molt més gran, la funció per a la inclusió automàtica de les noves dades, i la implementació d'una funció de la modificació de so per crear nous sons a partir del so recuperat.

Finalment, per acabar aquest apartat, s'estudiarà la separació cega de fonts (*BSS, Blind Signal Separation*). Aquest algorisme es basa en la determinació de les fonts originals quan només estan disponibles les sortides mesclades. Aquest tema s'introdueix en el projecte per permetre que, un cop s'hagi enregistrat la primera seqüència i s'estigui reproduint amb forma de *loop*, es pugui interpretar una segona seqüència i analitzar-la separadament del *loop* reproduït.

El model a seguir per descriure el problema el trobarem en (Cichocki, 2002). Aquest model s'ha anomenat *cocktail party*, ja que es basa en el disseny de sistemes intel·ligents d'adaptació i dels algoritmes d'aprenentatge associats que tenen habilitats similars als humans per centrar l'atenció en una conversa d'entre les moltes que estarien passant al mateix temps en una hipotètica festa de còctels.

Els éssers humans són capaços de concentrar-se en l'escolta d'una veu enmig d'altres converses i el soroll, però no tots els mecanismes d'aquest procés s'entenen completament. Aquesta capacitat d'escolta especialitzada pot ser degut a les característiques del sistema de producció de la parla humana, el sistema auditiu, o de l'alt nivell de processament de la percepció i del llenguatge.

En (Cichocki, 2002) es considera la situació en què els sons es graven en una habitació utilitzant una matriu de micròfons (Figura 2-8). Cada micròfon rebrà una còpia directa de la font de so -en algun retard de propagació basats en la localització de les fonts i el micròfon-, així com també diverses reflexions modificades -atenuada i retardada- i còpies de les altres fonts de so -que reboten en les parets i els objectes de l'habitació-.

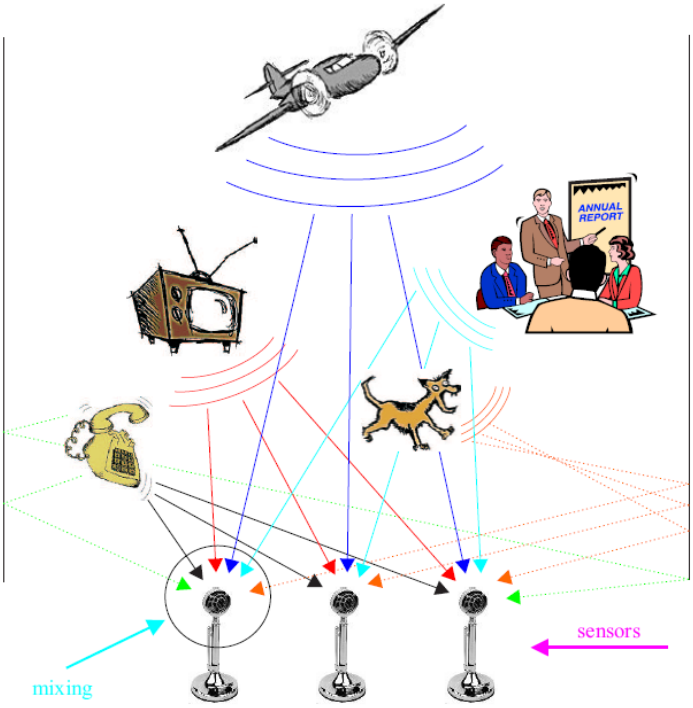


Figura 2-8. Il·lustració del problema del *cocktail party* (Cichocki, 2002).

Les distorsions dels senyals registrades depenen de les característiques de reverberació i l'absorció de la sala, així com dels objectes que hi ha dins de l'habitació. Aquestes es poden modelar com una resposta a l'impuls d'un sistema lineal, que proporciona un model de tots els camins possibles del so fins a arribar als micròfons.

Per esbrinar la seqüència original de la font caldrà cancel·lar o deconvolucionar la resposta impulsional de l'habitació. Com que no es té coneixement previ de com és aquesta resposta impulsional de la sala, s'anomena deconvolució multicanal cega o problema de la *cocktail party*.

Per tractar aquest problema de separació cega de senyals s'utilitzarà, com s'ha dit, l'algoritme *BSS*. Aquesta senyal de sortida és convolutiva ja que els senyals que es capten amb els micròfons són una suma amb eco i, per tant, serà necessària una deconvolució temporal no–lineal amb *BSS*. Tal i com s'exposa en (Harmeling, 2003), en un sistema d'aquest tipus s'observa una senyal de la forma (2.9). A partir d'aquí s'obté un algoritme base (*kernel methods*) per no-linearitats arbitràries invertibles.

$$\boldsymbol{x}[t] = \mathbf{f}(\boldsymbol{s}[t]) \qquad (2.9)$$

El vector de sortida $\boldsymbol{x}[t] := [x_1[t], \dots, x_n[t]]^T$ i el d'entrada $\boldsymbol{s}[t] := [s_1[t], \dots, s_n[t]]^T$ són matrius *n x 1*, i la funció **f**: $\Re^n \to \Re^n$. Llavors la idea passa per trobar la forma més acurada de la base ortonormal del subespai **f** tal com es mostra en (2.10).

$$\Xi = \boldsymbol{\Phi}_v(\boldsymbol{\Phi}_\mathbf{v}^\mathbf{T}\boldsymbol{\Phi}_v)^{-1/2} \qquad (2.10)$$

La matriu $\boldsymbol{\Phi}_v := [\boldsymbol{\Phi}(\boldsymbol{v_1}) \cdots \boldsymbol{\Phi}(\boldsymbol{v_d})]$ constitueix una base per a l'espai de dels vectors $\boldsymbol{\Phi}_x := [\boldsymbol{\Phi}(x) \cdots \boldsymbol{\Phi}(x_T)]$ de tal manera que l'espai vectorial generat és el mateix (2.11). L'algorisme proposa recuperar la senyal d'entrada $\boldsymbol{s}[t]$, tal com es mostra en (2.12).

$$\text{span}(\boldsymbol{\Phi}_v) = \text{span}(\boldsymbol{\Phi}_x), \qquad \text{rang}(\boldsymbol{\Phi}_v) = k \qquad (2.11)$$

$$\Psi_x[t] := \Xi^\mathrm{T}\boldsymbol{\Phi}(\boldsymbol{x}[t]) = (\boldsymbol{\Phi}_\mathbf{v}^\mathbf{T}\boldsymbol{\Phi}_v)^{-1/2}\boldsymbol{\Phi}_\mathbf{v}^\mathbf{T}\boldsymbol{\Phi}(\boldsymbol{x}[t]) \qquad (2.12)$$

En una sistema amb dos fonts com serà el cas d'aquest projecte, es consideren el monomis fins un cert grau com *quasi*-fonts. Així doncs, en aquest cas podríem tenir com a sistema de sortida una senyal del tipus (2.12)

$$\boldsymbol{q} = (s_1^2 s_2^2, s_1^2 s_2, s_1^2, s_1 s_2, s_1, s_2, s_1 s_2^2, s_2^2) \qquad (2.12)$$

Les bases es generen mitjançant polinomis de la forma $k(a,b) = (a^T b + 1)^n$. El senyal a analitzar pot ser de la forma (2.13).

$$\boldsymbol{x}[t] = \mathbf{A}(s_1[t], s_2[t])^T + \mathbf{c}(s_1[t] \cdot s_2[t]) \qquad (2.13)$$

Un cop obtingudes els vectors amb les bases corresponents, l'objectiu és identificar els senyals s_1 i s_2 entre els senyals extrets. La figura 2-9 mostra les gràfiques per de sortida i els vectors de la base que més correlació tenen respecte les senyals d'entrada.

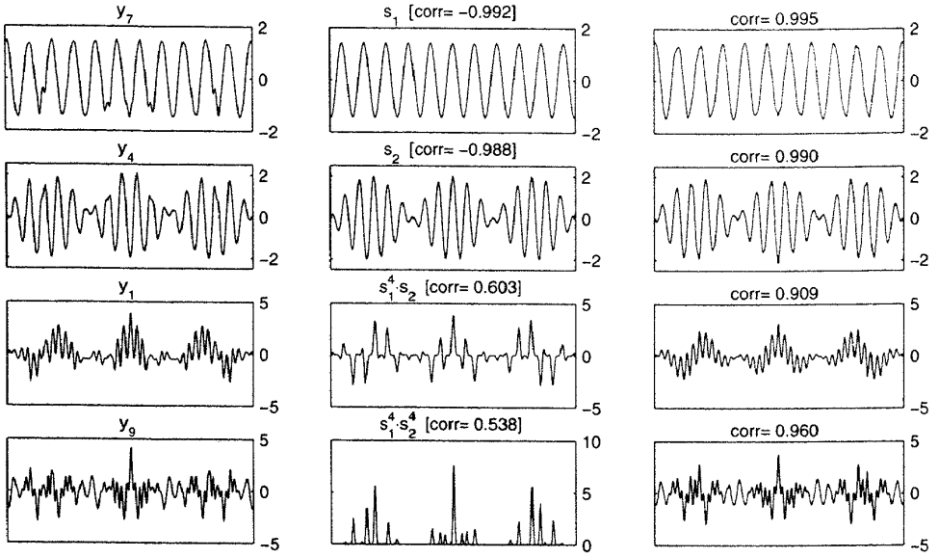


Figura 2-9. Senyals extretes i base de dimensió 4 (Harmeling, 2003).

La idea principal de l'algorisme rau en aplicar-lo dues vegades amb la mateixa base però en lloc d'utilitzar $\boldsymbol{x}[t]$ s'utilitza $\boldsymbol{y}[t]$. Ja que moltes de les senyals que no són útils són una combinació lineal de segon nivell o superiors, s'escalen perquè el seu màxim absolut sigui igual a 1. L'efecte del reescalat és que senyals amb grans pics són penalitzades, fent així que les senyals desitjades apareixin de nou amb correlacions més elevades després d'una nova desmodulació no-lineal.

$$x_1[t] = e^{-s_1[t]} - e^{-s_2[t]} \qquad (2.10)$$

$$x_2[t] = e^{-s_1[t]} + e^{-s_2[t]} \qquad (2.11)$$

En un dels experiments realitzats en aquest article es van mesclar dues senyals sinusoïdals $\boldsymbol{s}[t] := [s_1[t]\ s_2[t]]^T$ de freqüències diferents (2.10 i 2.11). Les bases de l'espai $\boldsymbol{\Phi_v}$ han estat generades polinòmicament mitjançant $k(x,y) = (x^T y + 1)^9$. D'entre tots els vectors generador de l'espai és redueix a 20 seguint l'exemple de la figura (2.9). El la figura 2-10 es mostra el resultat d'aplicar aquest algoritme.

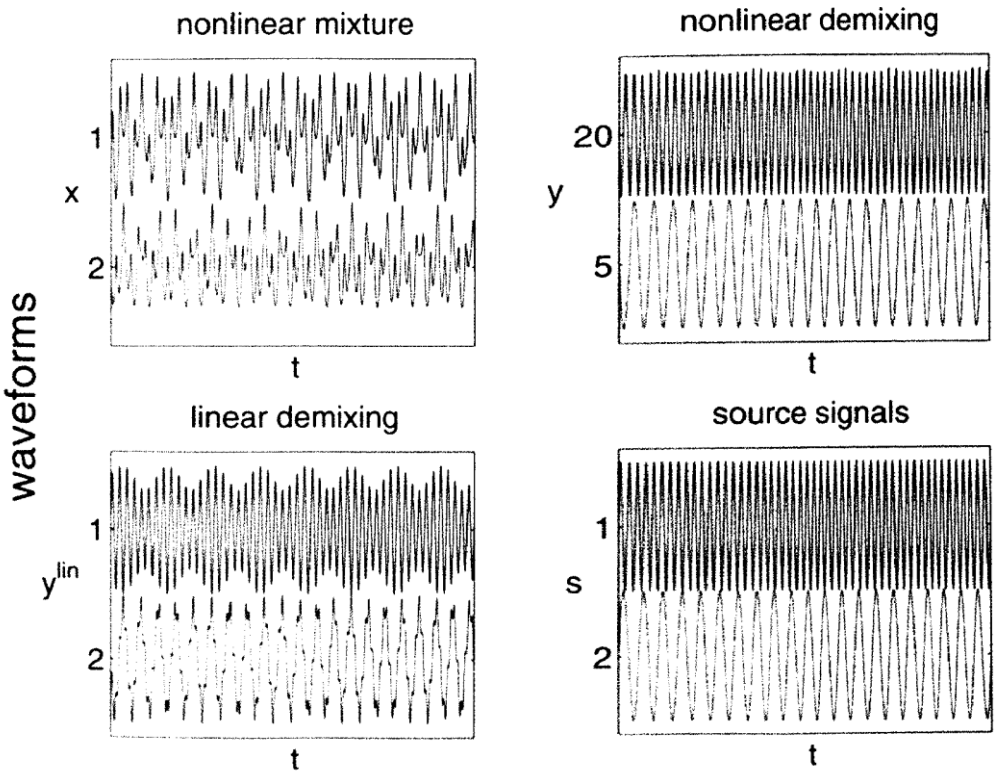


Figura 2-10. Senyals desmodulades amb base lineal i no-lineal (Harmeling, 2003).

2.4 MIDI

Un dels objectius més importants d'aquest projecte passarà per la conversió de senyals acústics a senyals MIDI. Aquests senyals no són més que instruccions que poden ser reconegudes per instruments compatibles. Contenen uns bits de capçalera, que són constants i aporten informació sobre quin camp es vol enviar, i després els bits del valor d'aquest, la longitud dels quals depèn del camp indicat per la capçalera.

MIDI (*Musical Instrument Digital Interface*) es comunica a una velocitat de 31.250 bps en sèrie. Les ordres MIDI són més sovint tres bytes de longitud, però de vegades més o menys. La taula 2-2 mostra aquestes instruccions.

Status Byte	Data Byte 1	Data Byte 2	Missatge	Llegenda
1000nnnn	0kkkkkkk	0vvvvvvv	Note Off	n=channel, k=key 0-127(60=middle C), v=velocity (0-127)
1001nnnn	0kkkkkkk	0vvvvvvv	Note On	n=channel, k=key 0-127(60=middle C), v=velocity (0-127)
1010nnnn	0kkkkkkk	0ppppppp	Poly Key Pressure	n=channel, k=key 0-127(60=middle C), p=pressure (0-127)
1011nnnn	0ccccccc	0vvvvvvv	Controller Change	n=channel, c=controller, v=controller value(0-127)
1100nnnn	0ppppppp	[none]	Program Change	n=channel p=preset number (0-127)
1101nnnn	0ppppppp	[none]	Channel Pressure	n=channel p=pressure (0-127)
1110nnnn	0fffffff	0ccccccc	Pitch Bend	n=channel c=coarse f=fine (c+f = 14-bit resolution)

Taula 2-2: Comandes MIDI i significat (Hass, 2010).

El primer byte s'anomena byte d'estat i conté tant el tipus de comandament MIDI i el nombre de canal: la zona baixa d'aquest byte (4 bits) és el canal, que s'utilitza per diferenciar els dispositius que estan en el mateix bus MIDI i varia de 0 a 15; la zona alta li diu al dispositiu MIDI quina instrucció ha d'executar. Podem trobar moltes instruccions tal i com es pot veure en la taula 2.1. En el cas que s'està estudiant, però, només en necessitarem bàsicament quatre: programa, nota, volum i Pitchbend. Aquest últim controla les petites desafinacions que es produeixen al vibrar una nota quan s'està tocant. Amb el clarinet, per exemple, aquestes es poden aconseguir movent una mica la boca cap a munt i cap avall o, menys habitualment, modificant la pressió de l'aire.

El segon i el tercer byte seran normalment el byte de to, que no és més que la nota que s'està reproduint i la seua velocitat, que és una forma de volum de la nota que s'està reproduint. En el cas del *Pitchbend*, però, que no apareixerà la nota, així doncs actuarà per a tot el rang d'una manera uniforme utilitzant els dos bytes per aquesta finalitat.

El volum es controlarà no per mitjà de la instrucció *Poly Pressure Key*, sinó que es modificarà amb la instrucció *Controller Change*. Finalment també s'utilitzarà aquest tipus de missatge per a la tria d'instruments. A l'annex 2 trobem un taula amb el instruments seguint la norma General MIDI 1.

2.5 Software

- **Matlab**

MATLAB és un entorn de computació numèrica i un llenguatge de programació. Permet manipular fàcilment matrius, dibuixar funcions i dades, implementar algorismes, crear interfícies d'usuari, i comunicar-se amb altres programes en altres llenguatges.

També compta amb un gran nombre de caixes opcionals (*Toolbox*), que ens permeten interactuar amb altres programes i dispositius. En el cas que ens ocupa, s'ha utilitzat la *DAC* (*Data Acquisition Toolbox*) que permet agafar la senyal directament de la targeta de so del sistema operatiu per analitzar-la.

- **Mmidi**

MMidi és una eina implementada per una Universitat alemanya (Bitzer, 2011) que permet enviar senyals MIDI directament des del Matlab a un port MIDI determinat. Segons els programadors, la latència que presenta és molt curta, per tant, és possible implementar un sintetitzador en Matlab. Així que un cop s'hagi analitzat el so amb Matlab es podrà enviar directament mitjançant aquest complement a un port MIDI.

Aquest *mmidi.dll* conté diverses funcions, les més importants seran:

- *mmidi ('show_devices'):* donarà sortida a una llista dels dispositius midi.
- *mmidi ('get_time'):* donarà sortida a una marca de temps d'un temporitzador, si un temporitzador està en marxa.
- *mmidi ('open_output', device, latència):* s'obre un flux de sortida midi, a la qual pot escriure. *device* és l'ID del dispositiu MIDI proporcionat per *show_devices*. La

latència és el retard afegit a cada marca de temps. Si la latència és 0, les marques de temps són ignorades.

- *mmidi ('close_output'):* es tanca un flux obert de sortida MIDI. Els fluxos oberts sempre han d'estar tancats abans que acabi el programa.
- *mmidi ('write_messages, inputMatrix):* escriu missatges MIDI per obrir la seqüència de sortida. Els missatges MIDI ha de tenir la següent sintaxi:
    - Cada missatge es compon d'una columna amb quatre files.
    - La primera fila és el temps del missatge.
    - La segona fila és el byte d'estat del missatge. El byte d'estat pot ser un nombre entre 64 i 127.
    - Les dues últimes files són els bytes de dades del missatge, entre 0 i 127.

- **Nuendo**

Aquest programa ens permetrà reproduir les senyals que MIDI que s'enviaran. És tracta d'un programari de música desenvolupat per Steinberg per a enregistrament de música, organitzar, editar i post-producció com a part d'una estació de treball d'àudio digital. Però a part d'això també pot servir per treballar amb temps real i reproduir amb tota mena d'instruments virtuals aquestes senyals MIDI.

- **MIDI Yoke**

MIDI Yoke és un controlador multimèdia MIDI, s'utilitza per connectar les sortides MIDI de Windows d'aplicacions per a qualsevol entrada d'altres aplicacions. Aquest serà el programa que farà de pont entre el Matlab i el Nuendo. Ens permetrà crear un port virtual d'entrada que captarà la senya que enviarem amb el Matlab i l'enviarà al dispositiu per fer sonar aquesta nota, és a dir que el flux de dades MIDI es passa directament de la sortida a l'entrada, per exemple:

[Seqüenciador de sortida 'matlab-Mmidi']==>

[sortida MIDI Yoke]==> [Entrada MIDI 'Nuendo']

Això et permet connectar la sortida MIDI d'un programa per a l'entrada MIDI d'un programa diferent. MIDI Yoke es pot configurar per proporcionar un nombre variable de ports MIDI (d'1 a 16). A més, cada port permet que obre múltiples d'entrada i de sortida: fins a 4 obertures per port.

3. Adequació de la senyal

3.1 Concepte general

La idea bàsica d'aquest apartat és la caracterització de la senyal enregistrada. Per a aconseguir això es proposa un sistema basat en l'enfinestrament temporal de la senyal en finestres de longitud 100ms. i en passos de 50 ms. A cada finestra se li aplica la funció correlació i s'obté un determinat *pitch*. Amb aquest procediment s'esbrina la nota i la seva duració, quan comença i quan acaba.

Un cop s'han captat aquests paràmetres es calcularan més detalladament elements més precisos de freqüència i volum mitjançant la implementació d'una transformada particular tenint com a referència la transformada Q de la doctora Brown. Aquí ja no s'utilitzaran els passos sinó finestres de 50 ms. directament per poder diferenciar més cadascuna.

A l'annex 8.1 es pot veure la taula d'equivalències entre les notes, la seua posició teòrica en la correlació de la senyal i el nombre de nota MIDI que es correspon. Val a dir que s'ha de considerar que la senyal s'ha mostrejat a 44100 Hz.

3.2 Captació de la senyal

Per a la captació del so s'utilitzarà una eina anomenada Data Acquisition Toolbox –DAC- per a Matlab; aquest complement ens permetrà captar la senyal d'àudio directament de la targeta de so.

L'esquema bàsic és mostra a la figura 3-1.

```
AI = analoginput('winsound');
addchannel(AI,1);
set(AI,'SamplesPerTrigger',L);
set(AI,'SampleRate',44100)
start(AI);
```

Figura 3-1: Codi per a captar el so de la targeta d'àudio.

En aquesta part es configura la captació de so tal com es pot veure explicat en el manuals Matlab sobre aquesta aplicació –*DAC*-. S'utilitzarà únicament un canal, amb una entrada mono es suficient. El valor *'SamplesPerTrigger'* vindrà donat pel temps o la

pulsació que es vol tenir. Finalment, s'introdueix la variable *'SampleRate'* que serà 44100 Hz i es comença a enregistrar.

En l'apartat 4.5, es veurà com es realitza exactament aquesta gravació, com interactua tot el sistema per enregistrar i enviar les instruccions necessàries.

3.3 Freqüència fonamental

Per a la detecció de la freqüència fonamental s'ha utilitzat la funció correlació normal, sense implementar la *Narrowed Correlation*, tal com s'explica en l'apartat 2.2. Degut a que tarda molt de temps en calcular-se no es òptima encara per a treballar en temps quasi-real. Per tant, es va fer servir la funció *xcorr de* Matlab, que proporciona la correlació en un temps de càlcul molt menor.

En aquest apartat es descriu, doncs, com s'ha detectat la freqüència fonamental de cadascuna de les finestres temporals en què dividirem la senyal (Figura 3-2).

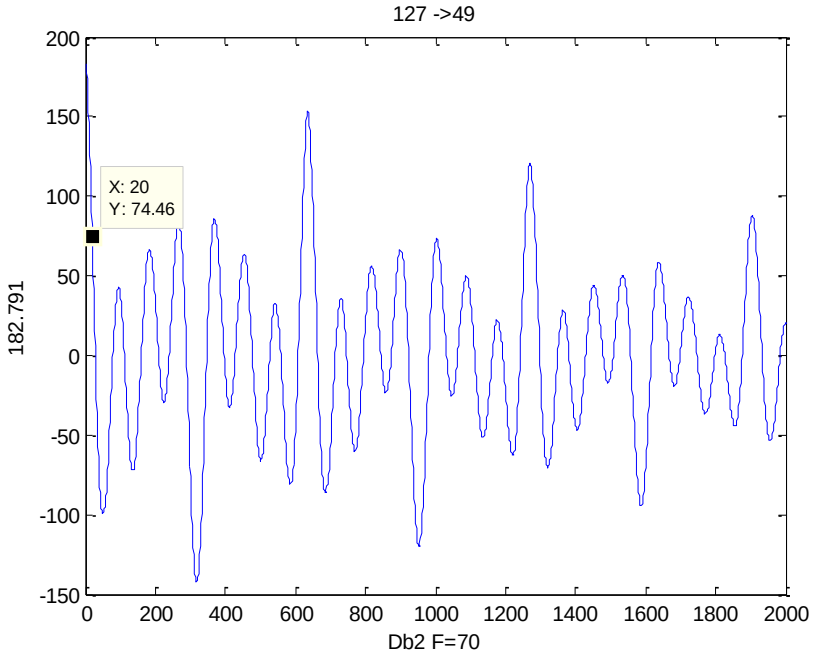
```
function [pitch,E] = correlacio(window,llindar,f0)
if nargin < 3                                          (1)
    f0=440;
    if nargin < 2
       llindar=0;
    end
end
x = xcorr(window,2000);
x = x(2001:4000);                                      (2)
E=x(1);
[~,B]=sort(x,'descend');
for i=2:length(x)
    if(abs(B(1)-B(i)))>20                              (3)
        if x(1)<2*llindar                              (4)
            pitch=0;
        elseif  x(B(1))>(2*x(B(i)))                    (5)
            pitch=1;
        else                                           (6)
            f=44100/(B(i)-1);
            pitch=round(12*log(f/f0)/log(2))+81;
            if pitch>103                               (7)
                pitch=1;
            end
            break;
        end
    end
end
```

Figura 3-2: Codi per a detectar la freqüència fonamental.

(1) Aquesta part del codi serveix per assignar la freqüència bàsica –*f0*- a la que sona l'instrument, per defecte serà 440 Hz. Però pot ésser modificada. També podem introduir el llindar de soroll que farem servir per discriminar.

(2) Es calcula la correlació de la senyal i s'agafa la segona meitat. Posteriorment ordenem el vector de major a menor, per esbrinar la posició del segon màxim.

(3) En aquest punt, es comprova que el segon màxim no sigui la cua del primer. Això es fa comprovant que estigui més enllà del posició 20. En l'exemple següent es com la cua, més enllà de la posició 20 -considerant notes- es troba normalment per sota del segon màxim. També comprovant que aquest 20 no sigui superior a la nota superior que volem captar, que segons la taula de l'annex 8.1, és la nota G6 a la posició 28 de la correlació.



Gràfica 3-1: correlació per a la nota més baixa del clarinet baix.

(4) En aquest punt es discrimina per soroll, si la correlació resultant és més petita que el doble del marge de soroll que s'obté es considera aquesta senyal com a soroll i es retorna un 0.

(5) Amb aquesta segona condició s'assegura que el segon pic segui prou alt perquè es pugui considerar que segurament es tracta d'una freqüència d'una nota interpretada. En aquest cas es retorna un 1 si no és compleix. La raó perquè no es retorna un 0 com el cas anterior és perquè amb el valor 0 actualitzarem el llindar de soroll cada vegada que es detecti.

(6) Aquest serà el cas en què s'ha detectat nota i s'aplica la fórmula tal com es fa a l'annex 8.1 per calcular l'alçada MIDI a què es correspon. Tot i això si va més enllà del registre del clarinet es retornarà un 1, sinó es retornarà el *pitch* de la nota.

3.4 Anàlisi de la senyal

Seguidament es descriu el programa principal per detectar les seqüències de notes. Primerament s'expliquen les variables i la seua inicialització.

- *y*: vector amb la informació àudio.
- *windowLength*: longitud de la finestra temporal que s'utilitzarà.
- *step*: passa amb què s'incrementarà la posició de la finestra.
- *window*: és la finestra auxiliar que s'utilitzarà per analitzar a cada pas la senyal.
- *notes()*: serà el vector on desarem la informació de les notes que analitzem. Aquest vector tindrà tres camps.
  - *notes(x,1):* contindrà la posició inicial
  - *notes(x,2):* contindrà la posició final.
  - *notes(x,3):* contindrà la posició MIDI.
- *llindar*: serà la mitja del soroll.

```
windowLength=floor(100*44.1);
step=floor(50*44.1);

numOfFrames = floor((n-windowLength)/step) + 1;
curPos=1;                                                    (1)
k=2;                                                         (2)
t=1;                                                         (3)
notes(1,1)=1;
notes(1,2)=1;
notes(1,3)=0;
llindar=1;
```

Figura 3-3: Codi per a captar el so de la targeta.

(1) *curPos* és el comptador s'utilitzarà per moure's dins la seqüència d'àudio enregistrada.

(2) *k* és l'índex del vector notes.

(3) *t* és la variable amb la qual controlar la mitja del llindar del soroll.

A continuació, es proposa el codi base del sistema (Figura 3-4). Un cop haver captat la senyal amb el micròfon es tracta de descriure quines notes s'han interpretat i entre quins marges de temps; com a primera idea es va agafant cada finestra i s'aplica la funció descrita a l'apartat anterior.

```
for j=S:numOfFrames
    window = (y(curPos:curPos+windowLength-1));
    [pitch,E]=correlacio(window,llindar,440);
    if pitch==0                                              (1)
        llindar=llindar*(t-1)/t+E/t;
        t=t+1;
    end
    if pitch~=notes(k-1,3)                                   (2)
        notes(k,3)=pitch;
        notes(k,1)=j;
        if k>1 && (j-notes(k-1,1))<minim                     (3)
            if(notes(k-1,3)~=0)
                if notes(k-2,3)==pitch                       (4)
                    k=k-1;
                elseif abs(notes(k-2,3)-                     (5)
                        notes(k-1,3))==1
                    notes(k-2,2)=j;
                    k=k-1;
                    notes(k,3)=pitch;
                    notes(k,1)=j;
                elseif abs(notes(k-1,3)-pitch)<3 ...         (6)
&& (j-notes(k-1,1))>1
                    notes(k-1,2)=j;
                    k=k+1;
                else                                         (7)
                    notes(k-1,3)=pitch;
                    notes(k-1,1)=j;
                end
            else                                             (8)
                notes(k-1,2)=j;
                k=k+1;
            end
        else                                  (9)
notes(k-1,2)=j;
            k=k+1;
        end
    end
    curPos = curPos + step;
end
```

Figura 3-4: Codi per a separar i captar les notes.

(1) Després d'haver aplicat la funció correlació es comprova si és soroll i en aquest cas s'actualitza el llindar. Amb aquesta fórmula es va actualitzant la mitja del soroll automàticament.

(2) Amb aquesta condició comprovem si hi ha algun canvi en la detecció del nou *pitch*. Si existeix es desa la nova posició inicial i la nova nota a *notes(k)*

(3) En aquest punt es comprova que la longitud de la nota anterior sigui més gran que 150 ms. per considerar-la nota. En cas contrari, es comprova també que aquesta senyal no es tracti de soroll. Si es tracta de soroll simplement el considerem com a tal. Contràriament ens podem trobar els tres casos següents:

(4) En el primer cas comprovarem es tracta de considerar que existeix vibrat que s'escapa durant unes mostres del *pitch* original (Figura 3-5).

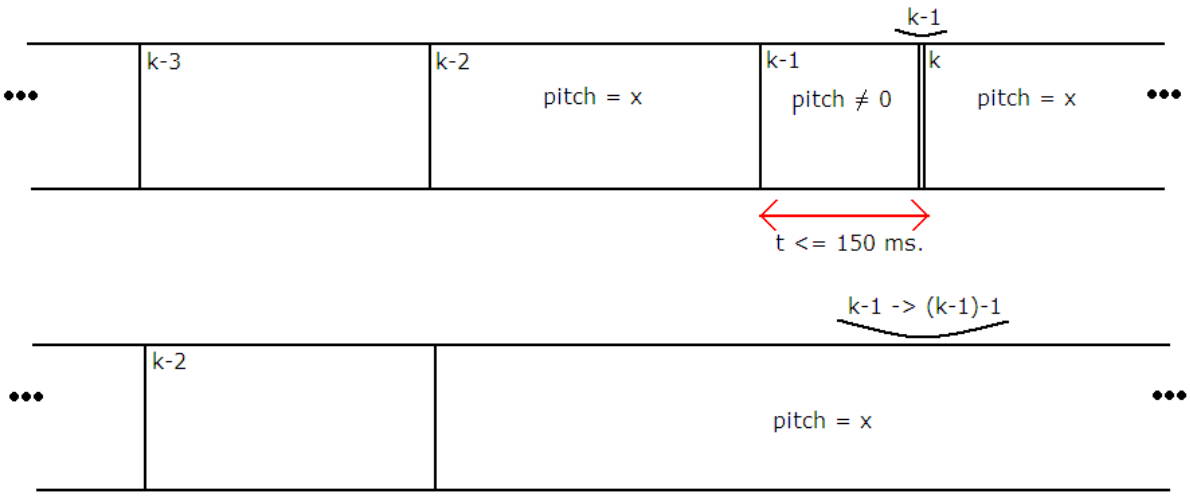


Figura 3-5: Petits canvis dins d'una mateixa nota.

(5) Aquest cas s'explica per si hi ha alguna nota que acaba en vibrat per dalt o per baix (Figura 3-6).

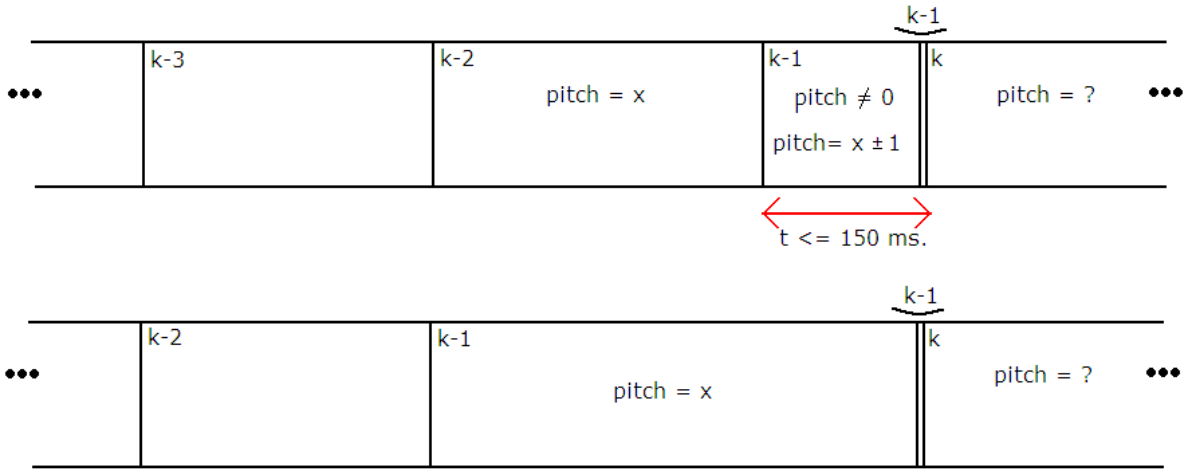


Figura 3-6: Petits canvis a final de la nota.

(6) En aquest cas es té amb compte si la nota curta està abans d'una altra nota separada un semitò o un to sencer. Si és així, es considera la nota com un ornament i s'afegeix al vector notes.

(7) Si aquest interval és soroll, es considera com una nota més i es desa.

(8) Finalment si la seqüència era més gran que 150 ms. es desa al vector notes (Figura 3-7).

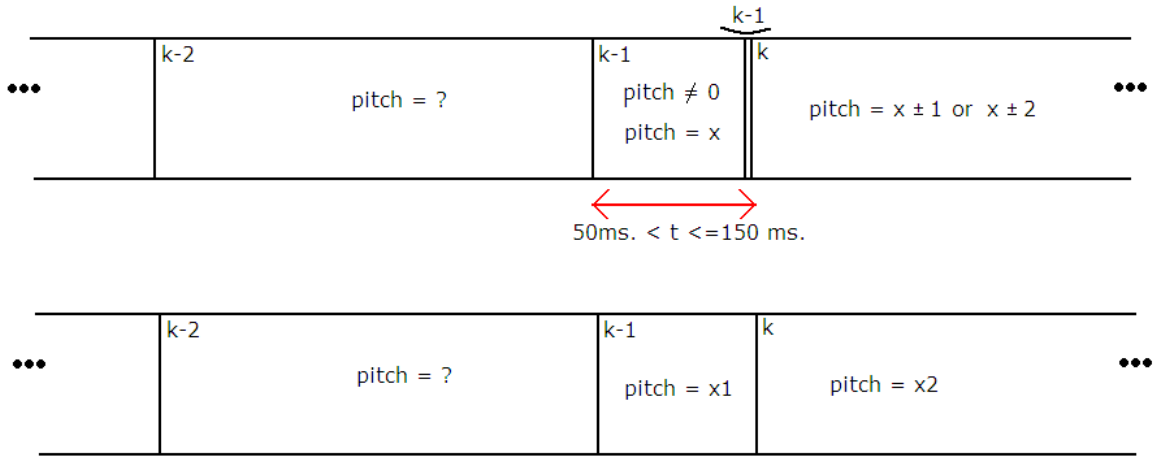


Figura 3-7: Nota de petita durada però que es considera.

(9) Si no és dona cap d'aquest tres casos, simplement es suprimeix la nota anterior.

Val a dir que aquesta variació de codi ha estat utilitzat per analitzar els arxius .WAV que s'han enregistrat per comprovar la viabilitat del sistema reconeixedor de notes. A l'annex 8.3 trobem unes taules que mostren com es detecten aquests canvis. A partir d'aquestes taules s'han derivat les condicions explicades. Aquestes taules mostren per a tota una seqüència de notes, concretament l'escala cromàtica amb tessitura de tres octaves, això es correspons a un total de 37 notes.

3.5 Variació d'amplitud i freqüencial

En aquest apartat, es presenta un codi perquè, una vegada s'hagin captat cadascuna de les notes, obtenir la caracterització de la freqüència i de volum. Això es farà seguint la idea dels articles de la Judith C. Brown, on parla de la transformada Q, descrita en l'apartat 2.2.

El codi implementat (Firgura 3-9) és una adaptació particular, ja que Brown allò que pretén és captar directament el *pitch* dividint l'espectre amb quarts de to.

En el cas present, encara que s'ha vist que el *pitch* es pot detectar amb prou èxit amb l'autocorrelació, mitjançant aquesta transformada es pot obtenir un resultat millorat per tal d'obtenir el paràmetre de la comanda MIDI *PitchBend*. Així doncs, ens permetrà detectar petites variacions d'afinació per simular el vibrat de l'instrument. Aquestes variacions es podran situar dins d'un nombre de subdivisions que seran regulars per qualsevol interval (Figura 3-8), com s'ha vist en l'apartat (Beauchamp, 2007).

Es tracta de, com que ja se sap la freqüència de la nota, situar-se dins de l'espectre entre un semitò per dalt i un semitò per baix. Centrat aquest espectre, es divideix amb parts iguales i es calcula la transformada en aquest punt. Cal remarcar que, per aplicar amb èxit aquest procediment, el número de mostres que s'haurà d'aplicar serà diferent cada iteració.

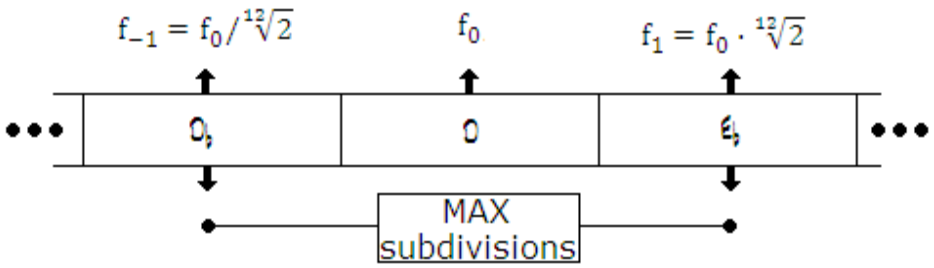


Figura 3-8: Representació de les divisions en dos semitons.

La figura 3-8 mostra els intervals on s'aplicarà aquesta idea. L'objectiu és poder captar el vibrat que s'aplica a una nota, amb això es pot enviar petites instruccions de canvis de freqüència i fer que la reproducció sigui més fidel. Per obtenir aquest resultat s'aplicarà allò explicat en l'apartat 2.2.

Primerament, es trobarà la freqüència amb la qual es començarà a treballar, tal com s'explica en l'annex 1. Amb (3.1) situarem la freqüència digital allí on s'hagi detectat amb la funció autocorrelació descrita en l'apartat 3.3. El valor del *Pitch* vindrà donat pel valor que s'hagi obtingut, el valor f_m serà la freqüència de mostreig i el valor f_{af} vindrà donat pel valor de freqüència que es vulgui donar a aquesta nota de referència (*Notaref*). Aquesta nota es tracta del la4, que normalment està afinada a 440hz. Tot i això es podrà canviar el valor, ja que hi ha vegades que interessa treballar a 442hz, en certes condicions de temperatura i humitat.

$$f_k = 2^{\frac{Pitch - Notaref}{12}} \cdot f_{af} / f_m \qquad (3.1)$$

A continuació, tenint en compte la implementació de l'algoritme amb Matlab es calcularà directament el número de punts amb els quals s'aplicarà aquesta transformada.

$$N[k] = \frac{1}{f_{k+1} - f_k} \qquad (3.2)$$

Per calcular aquest número de mostres ens situarem primer sobre la nota inferior per començar l'anàlisi, la freqüència relativa serà: $f_{-1} = f / \sqrt[12]{2}$. Un cop calculat aquest valor, que serà la primera de les variables f_k, s'obtindrà el valor f_{k+1}. Aquest valor dependrà del número de particions que interessi dividir aquest interval de dos semitons.

$$f_{k+1} = f_k \cdot 2^{2 \cdot DIV / 12} \qquad (3.3)$$

Llavors només caldrà definir aquest número de subdivisions i aplicar la transformada per cada punt concret (3.4). A cada pas es calcularan unes noves f_k, però s'aplicarà el mateix mètode.

$$X^Q[k] = \frac{1}{N[k]} \sum_{n=0}^{N[k]-1} x[n]\, e^{-2j\pi kn / N[k]} \qquad (3.4)$$

A continuació es presenta com s'ha programat aquesta funció que s'anomenarà Q (Figura 3-9). En aquesta programació no s'ha fet servir la finestra de Hamming ja que com que només interessa el pic no varia gaire el resultat.

```
function [E,p] = Q(y,pitch,faf,MAX)                (1)
f=2^((pitch-81)/12)*faf/44100;                     (2)
n=length(y);                                       (3)
f1=f/2^(1/12);                                     (4)
for i=1:MAX                                        (5)
    f2=f1*2^(1/(6*MAX));                           (6)
    N=1/(f2-f1);                                   (7)
    pos=f1*N;                                      (8)
    sum1=0;
    sum2=0;                                        (9)
    for j=1:n                                      (10)
        sum1=sum1+y(j)*cos(2*pi*j/N*pos);
        sum2=sum2+y(j)*sin(2*pi*j/N*pos);
    end
    x(i)=sqrt(abs(sum1)^2+abs(sum2)^2)/N;          (11)
    f1=f2;                                         (12)
end
[E,p]=max(x);                                      (13)
end
```

Figura 3-9: Transformada Q adaptada.

(1) Els paràmetres que rebrà aquesta funció seran la seqüència de mostres i el *pitch* de cadascuna de les nota que s'obtenen de la funció de l'apartat anterior. El paràmetre *f0* serà aquest valor de la funció correlació. Finalment el paràmetre *MAX* serà el nombre de subdivisions en què dividirem l'interval.

(2) Es troba la freqüència relativa de la nota amb la qual es treballarà, tal com s'explica en l'annex 1.

(3) La variable *n* contindrà la longitud de la nota que s'haurà captat.

(4) Se situa la freqüència a la qual es començarà a treballar. Aquesta serà un semitò per baix de la nota que s'ha detectat.

(5) Es crea el bucle que calcularà aquesta mena de transformada de Fourier particular que contindrà *MAX* mostres.

(6) Se situa el segon marge que estarà a una distància d'un to $f_2 = f_1 \cdot \sqrt[6]{2}$, més en nombre de subdivisions que interessin. $f_2 = f_1 \cdot \sqrt[6*MAX]{2}$.

(7) A continuació, es calcula el nombre de punts que tindrà aquesta particular transformada. Ja que sabem que cada posició ha d'estar separada la distància anteriorment anomenada, es força que estigui situat en aquell punt.

(8) Seguidament, se situa la posició que es vol calcular dins aquest número de mostres.

(9) S'inicialitzen els valors dels sumatoris per al sinus i el cosinus. Aquest s'utilitzaran per calcular la transformada (3.4).

(10) Es calcula la transformada però solament per a la posició que ens interessa.

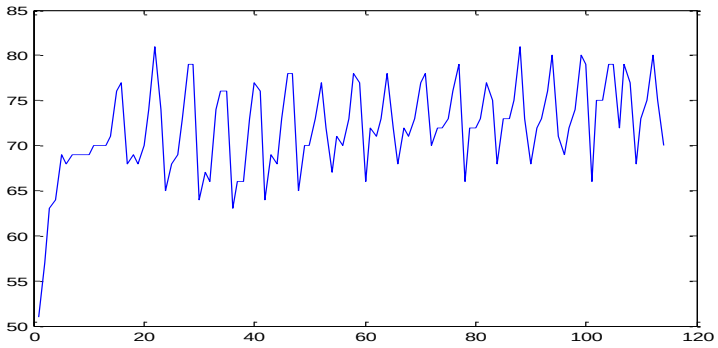
(11) Finalment es calcula el valor absolut d'aquesta transformada i es normalitza.

(12) Per passar al següent punt, es copia el valor segon al primer, per tornar al pas (6).

(13) Finalment es desa el resultat de la posició de la freqüència màxima i el valor d'aquest pic.

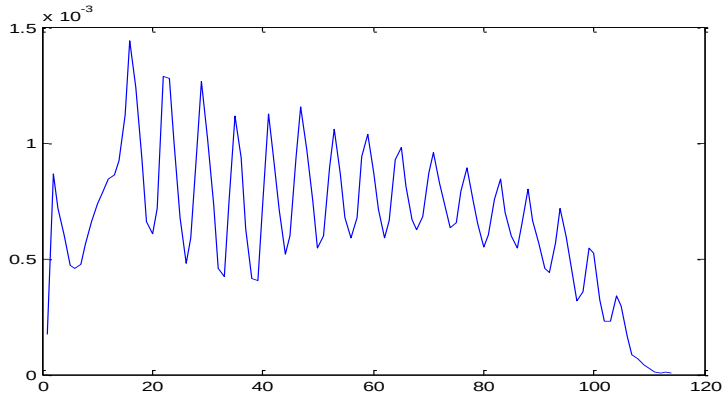
Amb aquesta funció, doncs, es pot caracteritzar tant els petits canvis de freqüència que es produeixen dins d'una mateixa nota, així com també les inflexions de volum que es poden produir quan es fa sonar un instrument. El valor de la funció *MAX*, tenint en compte que les instruccions MIDI són de 8 bits, es considerarà una subdivisió de 128 parts.

Com a primer exemple s'analitza un segment de nota obtingut del fitxer vibrat amb el clarinet baix. Es tracta de la nota D4. S'ha desat cada resultat en un vector i després s'ha dibuixat a una gràfica.



Gràfica 3-2: variació de freqüència en vibrat amb clarinet.

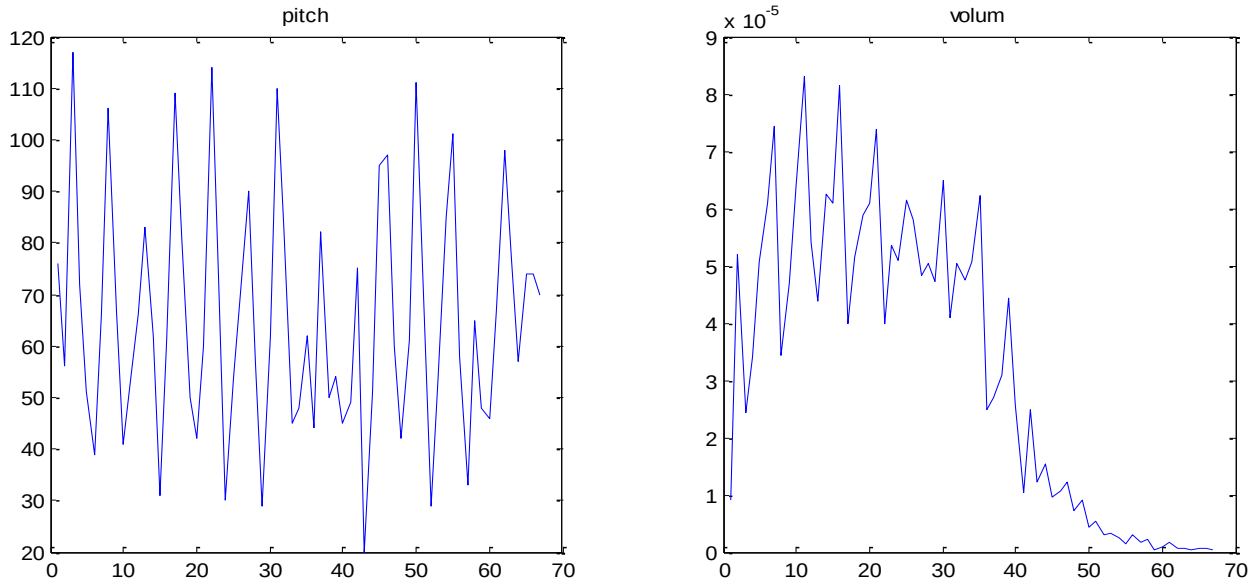
En aquesta gràfica 3-2, es pot veure fàcilment les petites variacions de freqüència, es compten sobre uns 18 cicles.



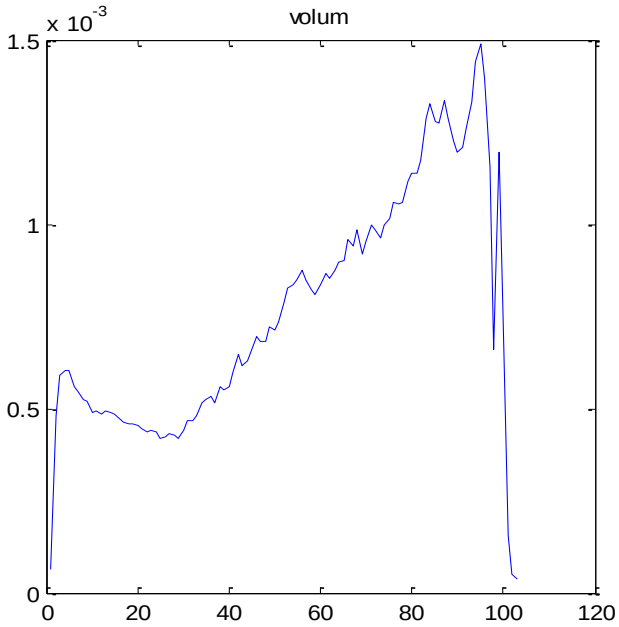
Gràfica 3-3: variacions de volum en vibrat amb clarinet.

La gràfica 3-3 es correspon amb les variacions en el volum. Aquí es pot observar com van lligats el vibrat amb el volum del clarinet. Al baixar la freqüència el que es fa és relaxar més l'embocadura i, per tant es perd una mica de volum.

Tot seguit comparem les gràfiques amb senyals generats MIDI per si tenen una detecció similar. La senyal MIDI s'ha generat amb clarinet virtual del VST Edirol Orchestral, amb la instrucció de controlador *Modalution* al màxim. Els resultats aquesta transformada doncs, són prou bons ja que ens permet veure el vibrat tal i com és. També corrobora que els la variació de freqüència va acompanyada d'una variació d'amplitud.

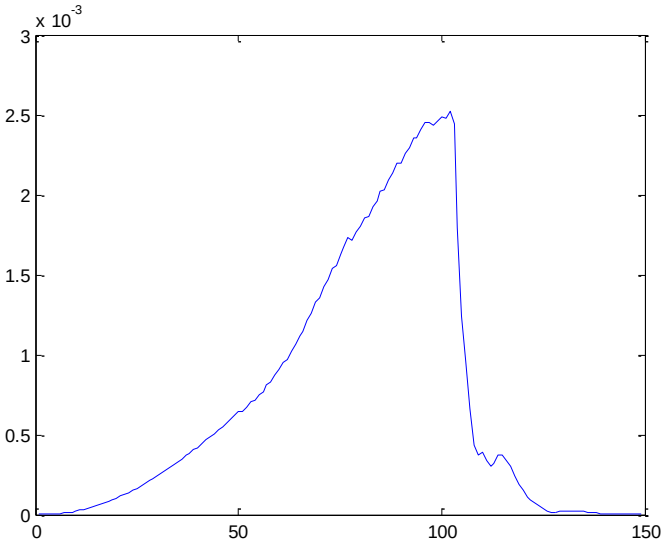


Gràfica 3-4: variació de freqüència i volum d'una nota MIDI amb modulació.



Gràfica 3-5: variació de volum d'un *crescendo* amb clarinet.

Posteriorment, s'han analitzat els crescendos o canvis de volum tant amb el clarinet com amb el so provinent d'un instrument MIDI. En gràfica 3-5, com ha estat enregistrat amb un clarinet, no es veu molt bé si l'evolució es ben bé lineal, exponencial... Ara bé analitzant el so que s'extreu d'enregistrar un senyal MIDI (Gràfica 3-6) que fa una corba de volum constant –amb el programa *volum*-, es no és lineal sinó més exponencial.



Gràfica 3-6: variació de volum d'un *crescendo* MIDI.

3.6 Resultats obtinguts

Una vegada implementat el codi bàsic per a la captació del to, es presenten els resultats que s'han obtingut quan s'han analitzat enregistraments. En l'annex 9.3 es poden veure dos seqüències vibrades amb el clarinet soprano i baix. Es mostren les taules de tots els canvis que es produeixen i com l'algoritme proposat ho resol.

La taula 3-1 mostra una sèrie de gravacions que s'han enregistrat per veure la viabilitat del detector proposat. Aquestes s'han fet tenint en compte els matisos de l'instrument. S'han gravat les notes cromàtiques en tot el registre amb diferents tipus d'atacs, de durades i de vibrats. L'última fila mostra en número d'errors que s'han produït en cada fitxer analitzat.

El total d'aquests suma 46 notes incorrectament detectades o no detectades d'un total de 703 (93%). Un cas extrem s'ha detectat per quan es toquen ràpidament notes molt curtes on la detecció baixa prou en picat (60%).

Així doncs els resultats han estat prou acceptables. El punts de màxim problema els trobem amb algunes notes molt greus, però sobretot amb els aguts. Això és degut a què quan, per exemple, es vibra amb una nota aguda l'afinació canvia molt més ja que el tub és molt més petit i els canvis afecten molt més.

A part d'això també cal dir que el pic de correlació a freqüències altes cada cop es va apropant més cap a 0, fet que provoca que cada cop hi hagi menys divisions per nota i, conseqüent, més errors.

Notes tocades	LLigat piano	Lligat	Picat curt 1	Picat curt 2	Vibrat lent	Vibrat ràpid	Picat curt 3	Picat curt 4	Picat curt 5	Picat curt 6	Pucat curt 7	Lligat 2	Lligat 3	Mig 1	Mig 2	Mig 3	Mig 4	Mig 5	Vibrat rapid 2
62	62	62	62	62	62	62	62	62	62	62	62	62	62	62	62	62	62	62	62
63	63	63	63	64	63	63	63	63	63	63	63	63	63	63	63	63	63	63	63
64	64	64	64	66	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64
65	65	65	65	68	65	65	65	65	65	65	65	65	65	65	65	65	65	65	65
66	66	66	66	69	66	66	66	67	66	66	66	66	66	66	66	66	66	66	66
67	67	67	67	71	67	67	67	68	67	67	67	67	67	67	67	67	67	67	67
68	68	68	68	73	68	68	68	69	68	68	68	68	68	68	68	68	68	68	68
69	69	69	69	75	69	69	69	70	69	69	69	69	69	69	69	69	69	69	69
70	70	70	70	76	70	70	70	71	70	70	70	70	70	70	70	70	70	70	70
71	71	71	71	78	71	71	71	72	71	71	71	71	71	71	71	71	71	71	71
72	72	72	72	79	72	72	72	73	72	72	72	72	72	72	72	72	72	72	72
73	73	73	73	81	73	73	73	74	60	73	73	73	73	73	73	73	73	73	73
74	74	74	74	82	74	74	74	75	73	74	74	74	74	73	74	74	74	74	74
75	75	74	75	84	74	75	75	76	74	75	75	75	75	74	75	75	75	75	75
76	76	75	76	85	75	76	76	77	75	76	76	76	76	75	76	76	76	76	76
77	77	76	77	87	76	77	77	78	76	77	77	77	77	76	77	77	77	77	77
78	78	77	78	89	77	78	78	79	77	78	78	78	78	77	78	78	78	78	78
79	79	78	79	91	78	79	79	80	78	79	79	79	80	78	79	79	79	79	79
80	80	79	80	92	79	80	80	81	80	80	80	80	81	79	80	80	80	80	80
81	81	80	81	94	80	81	81	82	81	81	81	81	82	80	81	81	81	81	81
82	82	81	82	96	81	82	82	83	82	82	82	82	83	81	82	82	82	82	82
83	83	82	83	98	82	83	83	84	83	83	83	83	84	82	83	83	83	83	83
84	84	83	84		83	84	84	86	84	84	84	84	85	83	84	84	84	84	84
85	85	84	85		83	85	85	88	85	85	85	85	86	84	85	85	85	85	85
86	86	85	86		84	86	86	90	86	86	86	86	87	85	86	86	86	86	86
87	87	86	87		85	87	87	91	87	87	87	87	88	86	87	87	87	87	87
88	88	86	88		86	88	88	92	88	88	89	88	89	87	88	88	88	88	88
89	89	87	89		87	89	89	94	89	89	90	89	90	88	89	89	89	89	89
90	90	88	90		88	90	90	95	90	90	91	90	91	89	90	90	90	90	90
91	91	89	91		89	91	91	96	91	91	92	91	92	90	91	91	91	91	91
92	92	90	92		90	92	92	97	92	92	93	92	93	91	92	92	92	92	92
93	93	91	93		91	93	93		93	93	94	93	94	92	92	93	93	93	93
94	94	92	94		90	94	95		94	94	95	94	96	93	93	94	94	94	94
95	95	93	95		91	96	96		96	95	96	95	98	93	94	95	95	95	96
96	96	94	96		92	96	98		98	96	97	96		94	94	96	96	96	96
97	97	95	97		93	98				98	98	97		95	95	96	96	97	98
98	98	96	97		94							98		96	95	97	97	98	
		97	98		95									97	96	98	98		
		98			96									98	96				
					97										97				
					98										98				
teòric	0	2	1	14	4	1	2	6	2	1	1	0	3	2	4	1	1	0	1

Taula 3-1: Detecció de notes amb el clarinet soprà.

4. Implementació del Loop-Station MIDI

4.1 Concepte general

En aquesta part s'explica la implementació gràfica que s'ha dissenyat amb el Matlab per tal d'interactuar amb el nucli de processament que s'ha descrit en l'apartat anterior. A continuació es presenta la forma final que tindrà la interfície per tal d'implementar el Loop-Station.

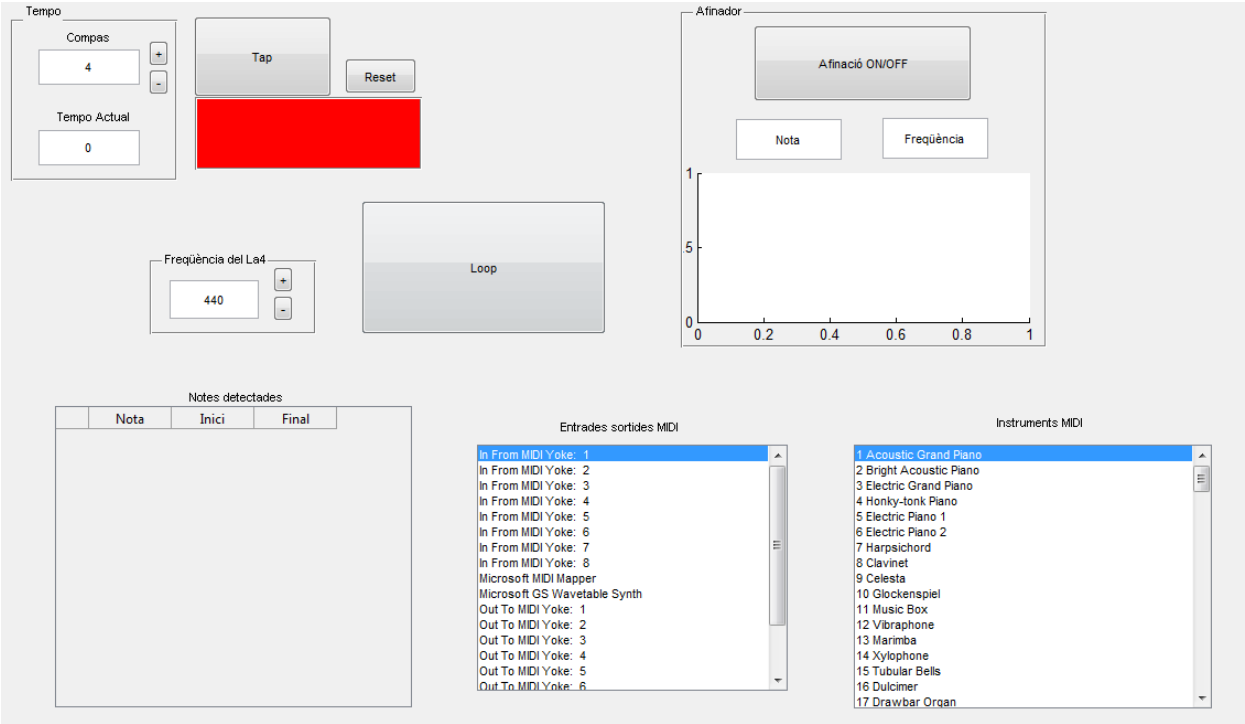


Figura 4-1: Presentació de la interfície gràfica.

En aquest esquema es diferenciaran tres parts bàsiques:

- A la part esquerra superior, la interfície proporciona un detector de tempo que servirà per marcar el temps, si es vol, durant la gravació. La icona 'Tap' servirà per marcar el *Temps* i *Reset* per tornar a començar. Podrem canviar el compàs, que servirà per controlar el nombre de mostres que agafem per modificar la pulsació, i a la finestra *Tempo Actual* ens apareixerà els *beats* per minut, és a dir, les pulsacions que hi ha per minut.

- La part central és el programa principal que s'utilitzarà per fer sonar el Loop-Station. Es podrà escollir la freqüència a central a la que s'analitzarà la senyal.
- La part dreta superior no serà més que un afinador precís utilitzant el codi proposat en l'apartat 2.5.
- A la part inferior apareixen tres taules. La taula esquerra mostrarà les notes que s'han detectat durant la gravació. La part central mostrarà els dispositius que es poden fer servir per treure les notes. Amb la taula de la part dreta es podrà escollir l'instrument virtual per a la reproducció.

4.2 Generació del vector MIDI

Un aspecte molt important per al funcionament de la interfície passarà per la codificació a senyals MIDI. Aquesta funció ens permetrà obtenir una estructura que es podrà enviar d'una manera molt directa al mòdul MIDI per a la seua reproducció. Tal com es descriu a la introducció, existeix un complement per a Matlab que envia directament les notes a una sortida MIDI. Aquest complement després de ser testejat, no ha donat bons resultats ja que s'encallava en moltes instruccions.

S'ha emprat així les comandes de Java especialitzades en MIDI ja que el Matlab pot importar aquest llenguatge. Amb aquest material, s'ha elaborat un vector d'estructures, que tindrà dos camps que serviran per controlar el temps d'enviament de la nota i el número d'aquesta nota.

- *Out (x).start*: temps relatiu de la nota que s'ha d'enviar respecte a l'anterior.
- *Out (x).note*: El camp de la instrucció MIDI a enviar, ja sigui informació de nota, volum o vibrat.

Aquesta funció tindrà com a paràmetres d'entrada la seqüència de la nota, la posició d'inici, la freqüència d'afinació del LA central perquè es pugui fer un anàlisi correcte i el número de contenidors en què dividirem els semitons per aplicar-hi la transformada tal com s'ha explicat en l'apartat 3.5.

El pas següent passarà per dividir la senyal en finestres i a cadascuna d'aquestes aplicar-hi aquesta transformada, que ens donarà una afinació i un volum concret. Llavors s'ha de considerar que per la primera finestra enviarem totes les senyals (P*itchbend*, de nota i de volum). Per a les següents finestres ja no s'enviaran senyals de nota.

```
function queueMidi(nota,v,inici,final,f0,N)                (1)
import javax.sound.midi.*                                  (2)
global out iMidi MAX MIN;
windowLength=floor(50*44.1);                               (3)
curPos=inici;
window=v(curPos:curPos+windowLength-1);
[e,p,~]=Q(window,nota,f0,N);                               (4)
aux=127*(e-MIN/2)/(MAX-MIN/2);                             (5)

if aux>127
    MAX=e;
    aux=127;
elseif aux<10
    aux=10;
    MIN=e;
end
out(iMidi).start= curPos/44100;                            (6)
out(iMidi).note = ShortMessage;
out(iMidi).note.setMessage(ShortMessage.PITCH_BEND,1,0,p);
iMidi=iMidi+1;
out(iMidi).start= curPos/44100;                            (7)
out(iMidi).note = ShortMessage;
out(iMidi).note.setMessage(ShortMessage.CONTROL_CHANGE,1,7,round(aux));
iMidi=iMidi+1;
out(iMidi).start= curPos/44100;                            (8)
out(iMidi).note = ShortMessage;
out(iMidi).note.setMessage(ShortMessage.NOTE_ON,1,nota-11,127);
iMidi=iMidi+1;
curPos = curPos + windowLength;

while curPos<final-windowLength                            (9)
    window=v(curPos:curPos+windowLength-1);
    [e,p,~]=Q(window,nota,f0,N);
    aux=90*(e-MIN/2)/(MAX-MIN/2)+37;
    if aux>127
        MAX=e;
        aux=127;
    elseif aux<0
        aux=0;
        MIN=e;
    end
    out(iMidi).start= curPos/44100;
    out(iMidi).note = ShortMessage;
    out(iMidi).note.setMessage(ShortMessage.PITCH_BEND,1,0,p);
    iMidi=iMidi+1;
    out(iMidi).start= curPos/44100;
    out(iMidi).note = ShortMessage;
out(iMidi).note.setMessage(ShortMessage.CONTROL_CHANGE,1,7,round(aux));
    iMidi=iMidi+1;
    curPos = curPos + windowLength;
end
out(iMidi).start= curPos/44100;                            (10)
out(iMidi).note = ShortMessage;
out(iMidi).note.setMessage(ShortMessage.NOTE_OFF,1,nota-11,127);
iMidi=iMidi+1;
end
```

Figura 4-2: Codi per captar el so de la targeta.

(1) La funció rebrà com a paràmetres la nota detectada, el vector de mostres enregistrat, la posició d'inici i final i la freqüència d'afinació.

(2) S'importa el mòdul Matlab que ens permetrà crear les instruccions MIDI. Es declaren les variables globals.
   - *Out* serà el vector explicant anteriorment.
   - *iMidi* és l'índex d'instrucció que es desaran. També s'utilitzarà com a total d'instruccions a l'hora de reproduir la seqüència.
   - *MAX* i *MIN* seran els valor màxim i mínim de l'energia de la transformada Q. Com s'ha dit és important que s'executi l'eina afinador abans de començar per situar-los al seu nivell corresponent.

(3) S'escolleix el valor de finestra (50 ms.) i se situa el valor de la posició a analitzar (*curPos*) al valor d'inici del vector.

(4) En aquest punt és calcula la transformada particular Q de la finestra, obtenint-se el valor d'energia i la posició d'afinació de la finestra.

(5) Se situa l'energia entre els marges de la instrucció MIDI. Si es més petit o més gran que els marges s'actualitza el resultat. Com es veia en la gràfica 3-6, la corba tenia forma més exponencial que lineal. Val a dir que la funció es considera lineal, ja amb que aquesta aproximació ja es donen prou bons resultats sonors.

(6) En aquest punt es desen els valors de posició al vector i es crea un missatge al camp *note* per desar-hi la informació del *pitchbend*. A continuació s'incrementa l'índex de nota.

(7) Es fa el mateix procés per desar la informació de volum.

(8) Un cop s'ha desat a informació de volum i d'afinació ja es pot escriure la nota perquè es pugui reproduir amb les característiques correctes.

(9) Fins que la posició actual sigui més petita que la posició final menys la finestra, s'iterarà el procés descrit (4), (5), (6) i (7). La nota no és tornarà a enviar.

(10) Finalment com a última instrucció enviar el final de nota (*noteoff*) amb la mateixa idea que les altres.

4.3 Loop central

En aquest apartat s'explicarà com s'ha programat aquest bucle principal. Es tracta de la funció *Loop_Callback* de l'annex 8.6, que controlarà el polsador *Loop* de la interfície.

```
function Loop_Callback(hObject, eventdata, handles)
global mode lLoop tLoop AI out iMidi nMidi              (1)

if mode==0                                              (2)
    tic;                                                (3)
    AI = analoginput('winsound');                       (4)
    addchannel(AI,1);
    set(AI,'SampleRate',44100);
    set(AI,'SamplesPerTrigger',inf);
    start(AI);
    mode=2;                                             (5)

    set(hObject,'BackgroundColor','green');
    main(handles);                                      (6)

elseif mode==2                                          (7)
    tLoop=toc;                                          (8)
    lLoop=round(tLoop)*44100;
    mode=3;                                             (9)
    set(hObject,'BackgroundColor','red');
    tic;

elseif mode==3
    mode=0;
      (10)
    set(hObject,'BackgroundColor','blue');
end
```

Figura 4-3: Codi central del programa.

(1) Es declaren les variables globals que permetran interactuar amb el bloc de processament central de l'apartat 3.2.

- *Mode*: és la variable que controlarà l'estat de la interfícies
  - *0*: és l'estat de repòs.
  - *1*: estat d'afinació. En aquest està s'estarà executant la funció *Afina_Callback* (apèndix 8.6). Apareixerà el polsador Afinació en verd i en la gràfica apareixerà el resultat.
  - *2*: estat de gravació. S'enregistrarà la seqüència.
  - *3*: estat de reproducció. Donarà la ordre perquè s'aturi la gravació, s'acabaran de processar les dades i s'enviaran llavors les notes midi.
- *lLoop, tLoop*: longitud en mostres i en segons respectivament del vector de so captat.

- *AI (Audio in)*: és l'objecte amb el qual podrem interactuar amb la targeta de so de *Windows* per captar les dades.
- *out*: és el vector que contindrà les instruccions MIDI a executar.
- *iMidi, nMidi*: són els comptadors d'instruccions i el número total d'instruccions enviades respectivament.

(2) Si el programa està en repòs i s'executa la funció *LOOP*, es procedeix amb l'enregistrament.
(3) Amb la funció *tic*, s'inicia el comptador que ens permetrà calcular el temps d'enregistrament.
(4) En aquesta part es configura la captació tal com s'explica en l'apartat 3.2.
(5) A continuació es canvia la variable *mode* per indicar que s'està enregistrant i es canvia la pantalla a color verd perquè es pugui controlar fàcilment.
(6) S'executa la funció *main* que serà l'encarregada d'analitzar les notes.
(7) Quan es torni el prémer el posador Loop voldrà dir que ja s'ha enregistrat la seqüència que es volia i ja s'està a punt per enviar les notes MIDI per a la seua reproducció.
(8) Amb la funció *toc*, s'esbrinarà la longitud entre les dues pulsacions de la icona *Loop*. Es desarà aquest resultat amb mostres i amb temps.
(9) Posteriorment es canviarà el mode a l'espera i la pantalla a color vermell, indicador de reproducció, quan s'hagi acabat el processament de les dades.
(10) Per acabar l'execució del bucle es tornarà a pitjar el polsador *Loop* que canviarà la variable *mode*, deixant així que s'enviïn noves senyals i tornant a l'estat de repòs.

Val a dir que aquesta part es va intentar implementar amb un sistema de *timers* tal com s'ha fet amb el compàs, però es van estar fent mesures amb els diversos tipus i no funcionaven gaire bé. Així que s'ha optat per aquest sistema una mica més senzill però que funciona millor.

La funció de processament que interactuarà amb aquest bloc és la funció *main*, aquesta funció s'ha explicat àmpliament a l'apartat 3.4, però només la part central. A continuació s'exposa el codi que permetrà controlar la variable *mode* i l'enregistrament. Aquesta funció tindrà com a paràmetre el camp per actualitzar el requadre notes en la interfície gràfica.

```
function main(handles)
%Declarem les variables que ens permetran interactuar amb altres
processos.
global mode notes AI y k llindar fTreball lLoop iMidi out tLoop;

trig=2500;  %variable que controla el nombre de mostres de captació
windowLength=floor(100*44);  %tamany de la finestra, 100ms,
step=floor(50*44);  %tamany de cada pas, 50ms,
curPos=1;  %variable de posició de lectura.
recPos=1;  %variable de posició d'escriptura.
t=1;    % controla la mitja del llindar del soroll
k=2;    % l'índex del vector notes
w=2;    % l'índex de les deteccions (notes + soroll)
y=[]; %vector que conté l'enregistrament del Loop
notes=zeros(100,3); %vector que contindrà les notes
lLoop=Inf; %Longitud del Loop, s'inicia a infinit.
iMidi=1;   %Index d'instrucció MIDI.
out=[];    %vector amb instruccions MIDI.
iNota=1;  %Index de notes detectades.
Mnotes=[];  %Matriu amb les notes detectades per a la GUI.

while curPos+windowLength<lLoop
    if mode==2
        if (recPos+trig)<(toc*44100)                                    (1)
                y(recPos:recPos+trig-1)=getdata(AI,trig);
                recPos=recPos+trig;
        end
        pause(0.01);
    elseif mode==3                                                      (2)
        if recPos<lLoop
            y(recPos:lLoop-1)=getdata(AI,lLoop-recPos);
            stop(AI);
            delete(AI);
        end
        recPos=lLoop-1;
    end
    while curPos+windowLength<=recPos
            /***** codi de l'apartat 3.5 ******/                       (3)
    end
end
notes(k-1,2)=curPos;
for j=w:k-1                                                             (4)
    if notes(j,3)>1
        queueMidi(notes(j,3),y,notes(j,1),notes(j,2),fTreball,64);

Mnotes(iNota,:)=[notes(j,3),notes(j,1)/44100,notes(j,2)/44100];
        iNota=iNota+1;
    end
end
set(handles.TNotes,'Data',Mnotes);                                      (5)
guidata(handles.TNotes, handles);
toc                                                                     (6)
```

Figura 4-4: Programa principal de detecció.

(1) Si encara s'està en l'estat de gravació, es va gravant nous intervals marcats per la longitud de *trigger,* la condició ens assegura de què quan pitgem el polsador per començar la reproducció no haguem enregistrat més del compte.

(2) Quan ja s'ha canviat l'estat, es desaran totes les mostres que facin falta.

(3) En aquest punt va el codi de l'apartat 3.4, on es detecten les notes. Hi ha una petita modificació ja que a cada iteració, si hi ha notes per processar, es processen, tal com es fa en el punt (4).

(4) Quan s'ha acabat el processament s'envien les notes que queden per processar i es desen a la matriu de notes.

(5) En aquest punt s'actualitzaran les notes captades per mostrar-se en la interfície gràfica.

(6) Per últim es pot fer servir el comptador *toc* per comprovar quan de temps es tardarà en començar la reproducció de la seqüència MIDI. Aquest resultat es de detallarà en l'apartat 4.5.

A continuació es descriu la segona part d'aquesta funció *main* on s'enviaran les notes per a la seua reproducció (Figura 4-5). Aquí s'utilitzarà aquest complement importat de Java per enviar les notes.

```
tic;                                                                    (1)
import javax.sound.midi.*                                               (2)
global MidiPort MidiInstrument r                                        (3)
info = MidiSystem.getMidiDeviceInfo;                                    (4)
outputPort = MidiSystem.getMidiDevice(info(MidiPort));
outputPort.open;
r = outputPort.getReceiver;
nMidi=1;
note = ShortMessage;                                                    (5)
note.setMessage(ShortMessage.PROGRAM_CHANGE, 1, MidiInstrument, 1)
r.send(note, -1)
while mode==3                                                           (6)
    if nMidi==1                                                         (7)
        aux=out(nMidi).start;
    elseif nMidi<iMidi
        aux=out(nMidi).start-out(nMidi-1).start;
    else
        aux=tLoop-out(nMidi-1).start+out(1).start;
        nMidi=1;
        pause(0.001);
    end
    while toc<aux                                                       (8)
    end
    tic;                                                                (9)
    r.send(out(nMidi).note, -1)
    nMidi=nMidi+1;
    end
    outputPort.close;
```

Figura 4-5: Programa principal de reproducció.

(1) Aquesta primera instrucció inicia el comptador amb el qual es controlarà el temps d'enviament de notes.
(2) S'importa la llibreria de Java que s'utilitzarà.
(3) Es declaren les variables globals que s'utilitzaran. Concretament seran les variables que es rebran de l'interfície gràfica com són el port i l'instrument que es farà servir. A part es crearà una nova variable global *r* que servirà per si a meitat gravació es decideix canviar l'instrument, la interfície gràfica ho pugui fer directament.
(4) S'obren els dispositius MIDI i s'escull aquell que prèviament s'havia triat amb la interfície gràfica. S'obre el port i es posa el comptador a 1 per començar.
(5) Abans de començar a enviar les notes, es transmet la instrucció de l'instrument que s'ha escollit. Es crea una variable *Shortmessage*, que contindrà la informació necessària i s'enviarà.
(6) A partir es començaran a enviar les dades, amb l'ajuda del comptador *tic* i *toc*, es podrà saber quan s'han d'enviar. Aquest bucle enviarà seguidament les instruccions fins que es torni a pitjar el polsador *Loop* de la interfície gràfica, que canviarà la variable global *mode* i s'acabarà.
(7) En aquest punt es controlarà el temps d'espera mitjançant la variable *aux*. Primerament si és la primera, *aux* contindrà el seu temps absolut. En la següent condició, passarà a ser la diferència entre temps entre notes quan estem al mig. Per últim, si s'arriba a final del vector, es comptarà quan queda pel final sumant el temps del primer element i canviant el *nMidi* de nou a 1.
(8) Un cop sabut el temps d'espera, es farà un bucle fins que el comptador arribi a aquest valor. Llavors s'enviarà la nota, es tornarà a posar a 0 el comptador per la següent instrucció i s'incrementarà l'índex de notes. Finalment es tancarà el port que s'havia obert.

4.4 **Afinador**

Amb aquesta part de la interfície (Figura 4-6) s'ha dissenyat un afinador. En l'apèndix 8.7 es pot veure com ha estat implementada. Però a grans trets no fa més que agafar 100 ms. d'enregistrament i analitza, mitjançant la correlació i posteriorment implementació de la funció Q, les mostres obtingudes.

Quan l'usuari prem la casella "Afinació ON/OFF", es posa en verd i comença la captació de la senyal, durant aquesta es va mostrant en les caselles de sobre la nota que es detecta (mitjançant la correlació) i la freqüència (mitjançant la transformada). En la

Fig. 4.6 es mostra quan de separat està el pic respecte l'afinació de la nota que indica (de -128 a 128 sent 0 la freqüència central de la nota).

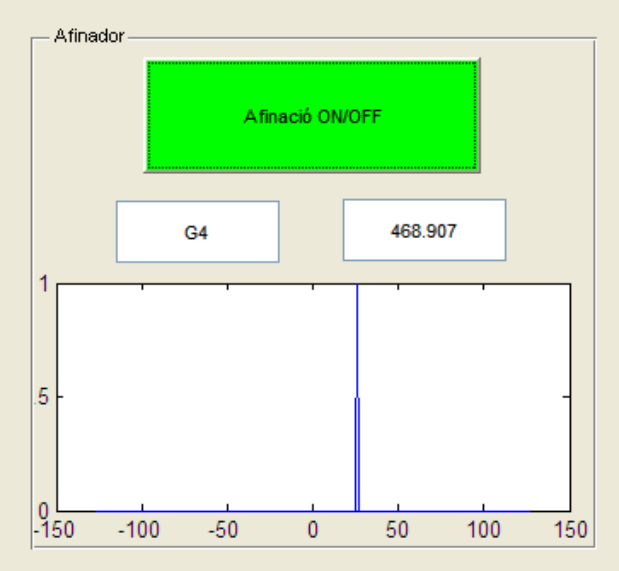


Figura 4-6: Afinador en funcionament.

Aquest apartat també serà important perquè s'utilitzarà per detectar el màxim i mínim de volum a què s'enregistrarà, amb això, es normalitzarà la seqüència detectada. A part, captarà el llindar de soroll amb què es descriminarà la funció correlació de captació i servirà per situar la freqüència del LA central.

És important que es torni a pitjar la casella per acabar la captació, deixant d'estar en verd. També remarcar que es pot aconseguir un afinador de tanta precisió com es vulgui, en aquest cas s'ha considerat que un marge de 128 divisions dins d'un semitò es suficient.

4.5 Captació del tempo

Aquesta eina vol ser una ajuda per al músic que estigui fent servir el programa. En un inici havia estat pensat per fer-se servir com a controlador de gravació, és a dir, que primer el músic introduís el nombre de compassos que vol enregistrar i amb la pantalla va seguir-los fins arribar al nombre de compassos desitjat. Finalment s'ha optat per fer-ho com un Loop-station normal *clicant* al començar i a l'acabar.

Això doncs la manera com funcionarà aquest complement és la següent:

- Mitjançant el la icona Tap l'usuari podrà anar introduint el tempo a què es vol obtenir la seqüència fins a un total del nombre de compassos multiplicat per dos. En aquest cas concret premeríem fins a 8 vegades la icona Tap per que el programa comencés a treballar. D'aquesta manera ens assegurem que el tempo que vol l'usuari segui una mitja més o menys regular.
- A la casella de "Tempo Actual" ens anirà apareixent el nombre de pulsacions per minut que li estem entrant de manera que l'usuari podrà comprovar si ell mateix és regular o no.
- Quan l'usuari ha pitjat el nombre total de compassos multiplicat per dos. La pantalla vermella canvia de color. Això indica que ja ens trobem al final de la seqüència de captació i que al següent canvi comença la seqüència fixa.
- La pantalla va canviant de color blau-negre regularment fins que es pitja el botó de *Reset*.

El codi utilitzat es troba en l'annex 8.8. Un aspecte important d'aquesta funció és la inicialització del temporitzador *timer* que es permetrà controlar el tempo de canvi de pantalla. Aquesta funció s'executarà cada vegada que aquest temporitzador s'activi i només canviarà el color de la pantalla.

4.6 Separació de font cega

En aquest apartat s'ha provat el codi proposat per (Harmeling, 2003) explicat en l'apartat 2.3. Aquesta funcionalitat del sistema s'ha implementat perquè, un cop s'estigui reproduint el senyal mitjançant les intruccions MIDI, l'usuari pugui enregistrar una nova seqüència. Aquesta nova seqüència interpretada estarà contaminada pel senyal provinent del Loop-Station. Es necessitarà llavors un algoritme de separació per obtenir una senyal lliure de senyals indesitjables.

Per provar el codi s'ha experimentat amb un sistema de captació basat amb dos micròfons que enregistren dues senyals de dos clarinets separats una distància de 3 m. Per una banda, un clarinet ha enregistrat una seqüència de notes curtes i picades ascendents i, per l'altra banda, s'ha enregistrat una melodia simple de tal manera que les dues senyals siguin totalment independents.

Les dues senyals enregistrades d'entrada són una mescla d'aquestes dues les senyals interpretades. Aquestes es mostren en la figura 4-7, on es veu que, en la gràfica inferior, predomina la senyal curta i en la superior la melodia.

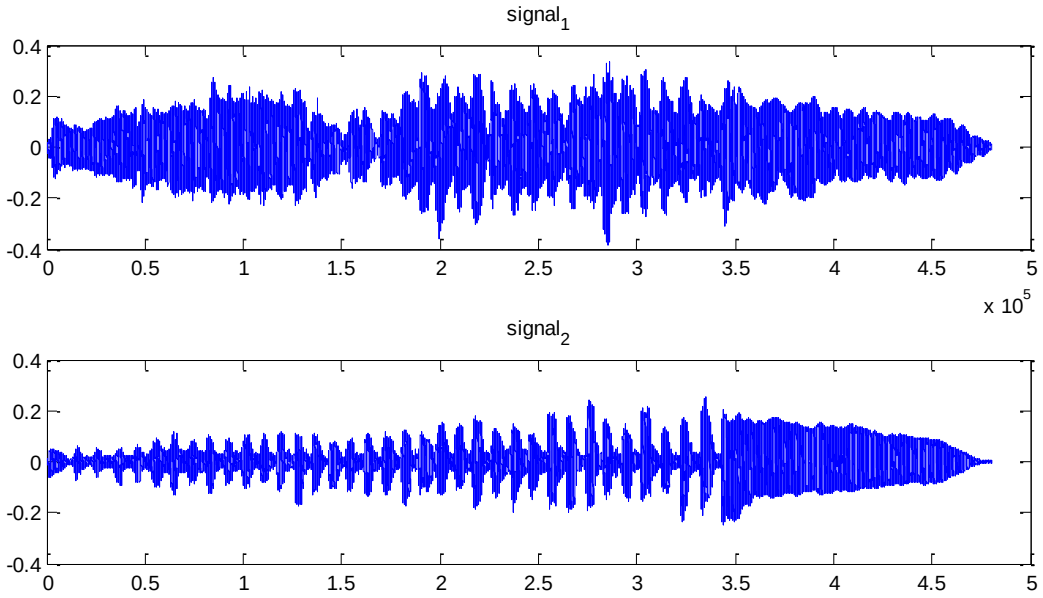


Figura 4-7: Seqüència enregistrada.

Posteriorment es processen aquestes senyals utilitzant un polinomi generador i després s'escolleixen els vectors generadors amb més correlació respectes les senyals. El procés de reducció de vectors generadors és repeteix dues vegades i després de la demolulació les senyals de sortida són els mostrats en la figura 4-8. Si bé el segon senyal recuperat mostra una major presència de la melodia simple que en l'original, el primer encara manté una barreja similar a l'original. Una possible explicació de tal efecte es pot atribuir a que la distància entre els instrumentistes resulta prou gran com per assolir una eliminació considerable en els senyals posteriorment processats.

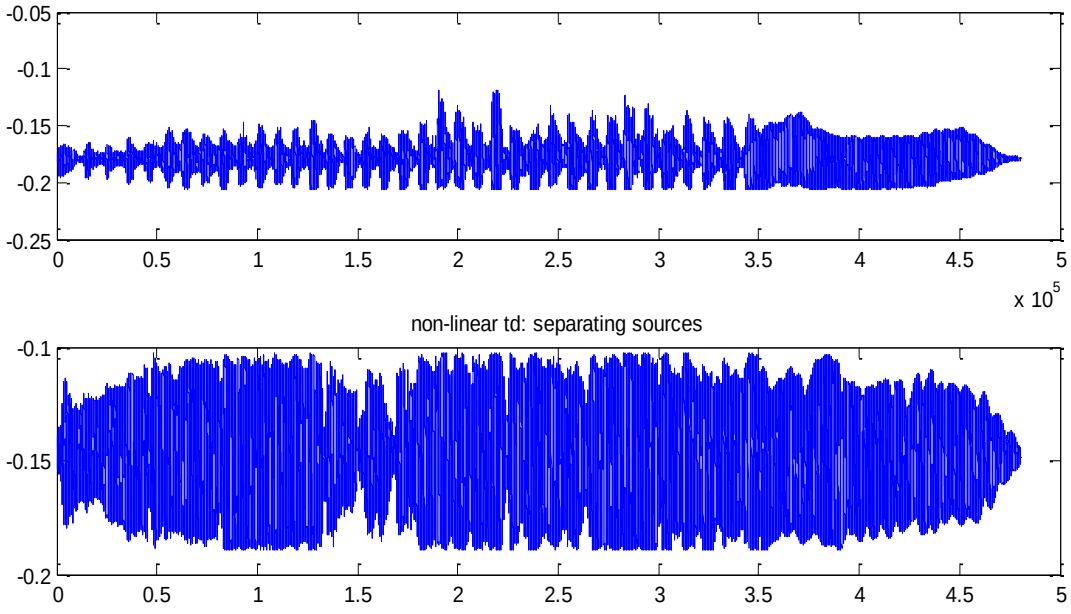


Figura 4-8: Seqüència separada.

4.7 Resultats obtinguts

Un cop implementat l'algoritme s'ha procedit a analitzar el codi sencer.

S'han tocat, com en els resultats anterior, notes de diferents maneres. Els resultats com són d'esperar són similars. Aquí es farà més èmfasi amb el resultat sonor i diferenciant cada tipus de nota. Per això es farà servir el quadre de notes que apareix en la part inferior esquerra del detector. Aquesta taula mostra el número de nota detectada i el seu principi i final.

- Notes lligades: En la seqüència de notes conjuntes sense respirar el detector detecta quasi totes les notes correctament, normalment detecta totes menys una o dos, mínim un 94%. Les notes que no detecta, però si que les interpreta ja que amb la transformada Q situa l'alçada de l'afinació al valor que toca. A mesura que es van tocant notes lligades més ràpidament la taxa de detecció, com és lògic, baixa.

- Notes picades: En les notes picades llargues el resultat és similar a les notes lligades. Les notes picades curtes resulten ser les més problemàtiques de detectar tal com s'havia comentat en l'apartat 3.7. En alguns dels casos, la detecció baixa fins un 60%, tot i que passa com el cas anterior i algunes de les notes si que sonen degut a què, amb l'algoritme, es consideren tot una, fent que el resultat sonor no sigui tant negatiu.

- Notes vibrades: Les notes vibrades es detecten bastant bé amb l'algoritme dissenyat ja que ha estat pensat bàsicament per a elles. Així doncs tenim una mitjana de detecció d'un 90%, el resultat és similar a allò obtingut en les seqüències de vibrat de la taula 3-1.

Pel que fa al funcionament del *loop-station* val a dir que no s'ha pogut aconseguir programar-lo de tal manera que, quan es premi el botó de final d'enregistrament per començar la reproducció, es faci automàticament. En pitjar-se la icona de parar l'enregistrament el programa calcular les notes que li fan falta per acabar la seqüència, que tal com s'ha programat no seran més de dues notes que quedaran. Tot i això aquest temps no supera normalment el segon.

En reproducció es podrà canviar l'instrument que està sonant mitjançant la taula inferior dreta. En l'annex 8.6 es troba el codi per a tota la interfície. Tant el codi de canvi

d'instrument com de canvi de port MIDI es troben en les funcions *instrMIDI* i *MIDIports* respectivament. Les seves crides -*callbacks*- són les encarregades de situar els valors de la variables globals respectives perquè, quan sigui el moment d'enviar les notes, es puguin fer servir els valors adequats. Per aquest motiu és necessari escollir el port MIDI que es farà servir abans de començar l'enregistrament.

Tot seguit es mostra un exemple on s'analitza una seqüència interpretada amb el clarinet (Figura 4-9) i el resultat que s'obté al passar-ho pel programa implementat.

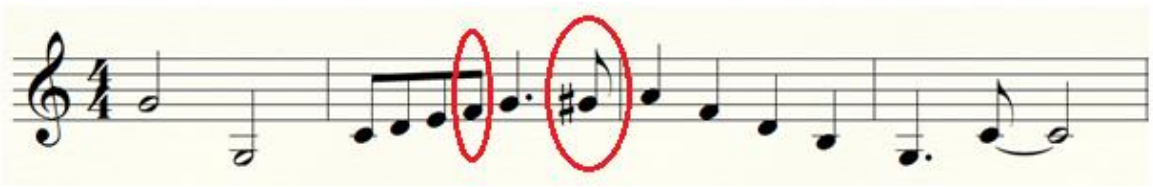


Figura 4-9: Seqüència interpretada.

Les notes detectades es mostren a la figura 4-10. Com es pot veure el sistema es deixa dues notes que no apareixen a la taula, tot i això acústicament el resultat es satisfactori ja que es consideren com una lleugera desafinació de la nota seguint l'algoritme explicat en l'apartat 3.3. Llavors amb la transformada implementada (Apartat 3.4) s'analitzen aquestes dues notes d'una manera conjunta però se situa cada interval de temps dins d'aquesta a la freqüència corresponent de manera que a l'enviar el senyal *pitchbend*, el resultat sonor és bo.

Notes detectades

	Nota	Inici	Final
1	77	1.2971	2.4445
2	65	2.4445	3.3923
3	70	3.4422	3.8413
4	72	3.8413	4.0907
5	74	4.0907	4.6395
6	77	4.6395	5.7370
7	79	5.7370	6.1361
8	75	6.1361	6.7846
9	72	6.7846	7.2336
10	69	7.2336	7.7823
11	65	7.8322	8.6803
12	70	8.6803	11.5238

Figura 4-10: Seqüència captada.

5. Implementació amb el clarinet baix real

5.1 Concepte general

La idea final del projecte era, una vegada s'ha estat capaç d'enviar les senyals MIDI, crear un dispositiu MIDI real. És a dir, amb l'ajuda del microcontrolador i un sistema basat en electroimants, una boca amb material termoplàstic, un compressor, una vàlvula de pressió i diversos complements electrònics, poder fer sonar un clarinet baix a través de les senyals MIDI corresponents. Aquest bloc s'anomenarà bloc d'actuació i aquests elements s'exposen en l'apartat 5.2 i es mostren en la següent figura (Figura 5-1). Val a dir que falta un element, com és pot suposar, que serà la font d'alimentació.

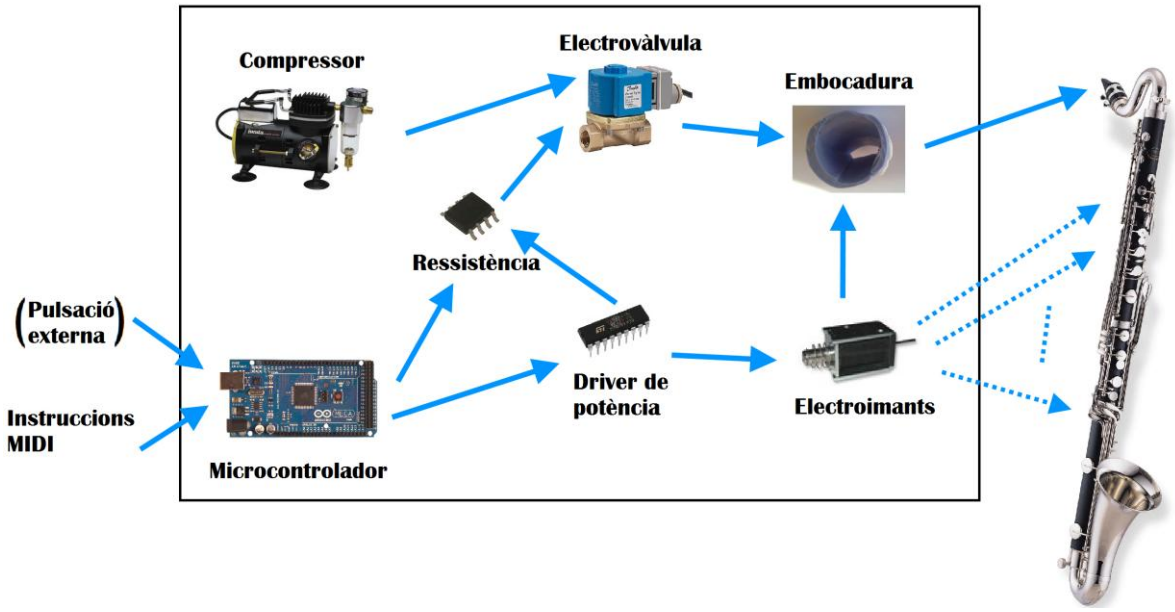


Figura 5-1: Parts que componen el bloc d'actuació.

Aquest esquema s'ha dissenyat inspirant-se amb els articles descrits en l'apartat 2.1 (Petersen, 2009) on explica la manera com s'ha realitzat el prototip del robot saxofonista WAS-1. La figura 5-1 és una simplificació de tot el mecanisme compacte que aquest equip ha dissenyat. Aquest mecanisme es pot veure en la figura 5-2. Es tracta d'una implementació molt acurada que té en compte tots els òrgans humans que intervenen en la producció de l'aire.

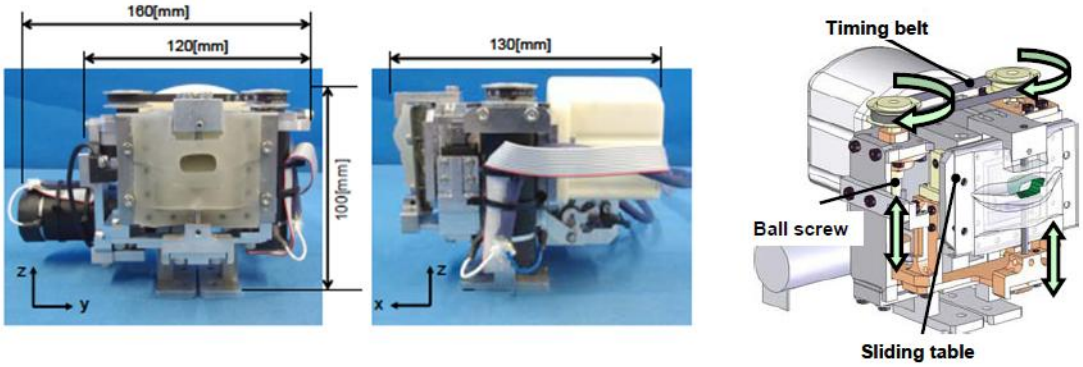


Figura 5-2: Figures i esquema de la part respiratòria del robot WAS-1 (Petersen, 2009).

El mecanisme de la llengua, per exemple, està realitzat amb un petit motor. En l'esquema proposat aquesta funció seria implementada per un electroimant. La figura 5-1 proposa un model implementar d'una manera més senzilla tot aquest treball. El *driver* de potència serà l'encarregat do suplir amb un voltatge adequat les bobines. Aquestes serviran, per una banda, per controlar la pressió de l'aire que ve del compressor (vàlvula) per l'altra, la pressió de l'embocadura i la llengua i, per últim, el mecanisme dels dits per prémer les tecles.

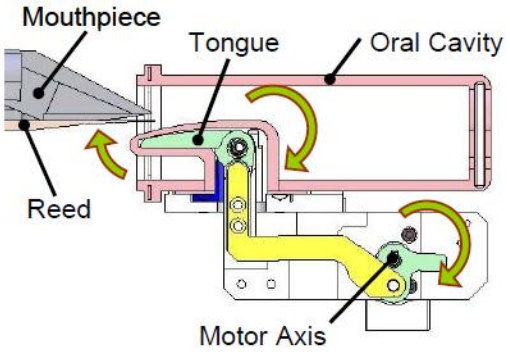


Figura 5-3: Esquema del mecanisme de la llengua (Petersen, 2009).

Per altra banda també s'ha estudiat com obtenir les notes MIDI sense necessitat d'emprar un ordinador, programant directament uns xips anomenats *DSP* –Processadors de Senyal Digital- seguint les investigacions de (Arvin, 2009). Aquesta part s'encarregarà de captar la senyal del micròfon i treure la senyal MIDI corresponent. Aquest bloc s'anomenarà bloc de captació i processament (Apartat 5.3).

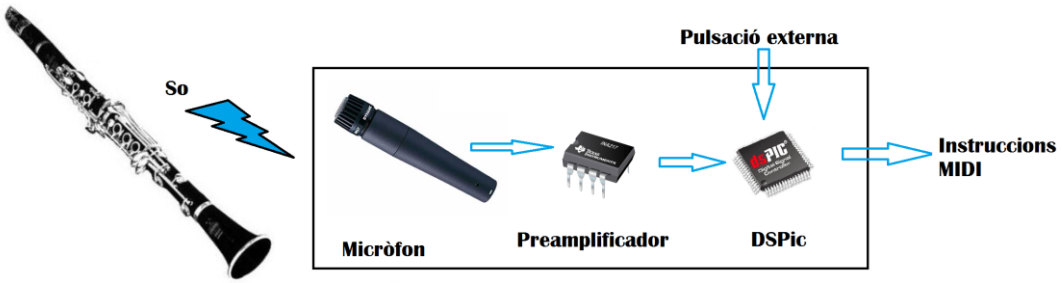


Figura 5-4: Parts que componen el bloc de captació i processament.

Aquesta distinció entre aquestes dues parts permet que els dos sistemes puguin treballar independentment sense necessitat l'un de l'altre. Per una banda podem obtenir un sistema que d'un so i un tempo determinat tingui com a sortida senyals MIDI en bucle que descodificaran allò que s'ha interpretat. I per l'altra banda, tindrem un sistema on es podrà entrar una seqüència MIDI i farà sonar mitjançant aquest elements (Figura 5-1) el clarinet baix. En la realització d'aquest projecte s'ha fet més èmfasi com s'ha pogut veure amb l'apartat de processament de la senyal (Apartats 2 i 3). A continuació, però s'explicarà com s'ha pensat i realitzat aquest bloc.

5.2 Bloc d'actuació

Primerament, es va estudiar el tema del microcontrolador per rebre les aquestes instruccions i descodificar-les. La placa escollida va ser l'*Arduino Mega* (Figura 5-5), basada en el xip *ATmega1280* de l'empresa *ATMEL*. Compta amb 54 entrades/sortides -de les quals 14 es poden utilitzar com sortides *PWM*-, 16 entrades analògiques, 4 UART (Transmissor-Receptor Asíncron Universal), un oscil·lador de vidre de 16 MHz, una connexió *USB*, un connector d'alimentació, una capçalera *ICSP*, i un botó de reinici.

Figura 5-5: Placa del microcontrolador *Arduino Mega.*

A part, aquesta placa també inclou alguns dels ports que es poden configurar com interrupcions. Amb un pedal es pot configurar l'inici i final del bucle que es vol enregistrar, ja que també compta amb quatre ports sèrie amb els quals pot comunicar-se amb tres dispositius diferents alhora, ja sigui l'ordinador –directament amb el cable USB-, un DSP (Processador de Senyal Digital)...

Mirant allò que es necessita amb el clarinet baix, es requereix d'un total de 19 electroimants per poder controlar totes les notes que es poden fer sonar. La idea és, utilitzant les sortides digitals de l'*Arduino*, controlar aquests electroimants amb l'ajuda del dispositiu de Texas Instruments *ULN2803A* (Figura 5-6). Aquest dispositiu permet obtenir per entrades de 5 i 0 volts, que són les que dóna l'*Arduino*, corrent nul a l'entrada i voltatges de fins a 20 volts a la sortida. Corrent nul d'entrada es convenient per no cremar el microcontrolador i voltatges de 20 V són suficients per obtenir la pressió necessària per prémer la tecla.

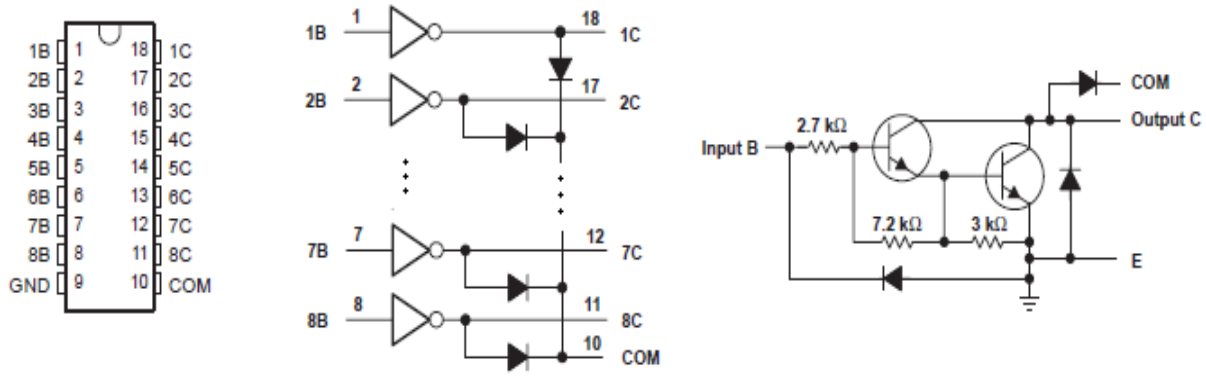


Figura 5-6: Vista general, diagrama esquemàtic general i particular de cadascun dels parells de Darlington.

El *ULN2803A* és una matriu de transistors Darlington d'alt voltatge i d'alta corrent. El dispositiu consta de vuit parells de Darlington PNP que compten amb sortides amb díodes pinça de càtode comú per a la commutació de càrregues inductives. La corrent de col·lector de cada parell Darlington és de 500 mA. Els parells Darlington es poden connectar en paral·lel per a una major capacitat de corrent.

Així doncs, amb l'ajuda d'aquest aparell es pot connectar les sortides analògiques de *l'Arduino* als ports d'entrada *xB*. Amb als ports de sortida (*xC*) només restarà connectar directament l'electroimant i aquest a la font d'alimentació. Necessitarem llavors un total de 3 xips d'aquest tipus, tenint un total de 24 canals on es farien servir 19 per controlar

les tecles del clarinet, dos més per controlar l'embocadura i un més per controlar la pressió com es veurà a continuació.

L'electroimant testejat ha estat el PED 44A series (Figura 5-7), el qual permet entrada de corrent continu. Amb 15-20 volts s'ha comprovat que fa la força suficient per prémer la tecla del clarinet. Com es pot veure en la següent figura té una molla amb la qual recupera l'estat inicial i és amb la banda posterior amb la qual es pitjaria la tecla.

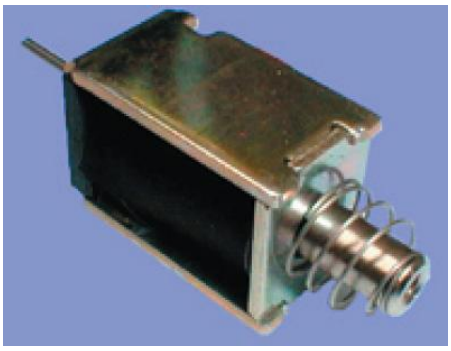


Figura 5-7: Electroimant PED 44A series.

Pel que fa al sistema de producció del so, es va demanar un pressupost a una empresa de material termoplàstic (Inlavi S.A) d'una mena de boca tal com van dissenyar l'equip de la Universitat de Waseda amb el saxofonista WAS-1, encara que la seva excedia el requisit de baix cost del prototip que ens havíem proposat. Llavors es va procedir a demanar al departament d'odontologia del Centre Mèdic Universitari de Bellvitge, si tenien algun prototip de boca, però no va ser possible, però es va aconseguir una silicona especial no tòxica molt fàcil de motlletjar, consisteix amb una base i un catalitzador que al mesclar-se donen una textura molt similar al llavi. A continuació es presenten els prototips creats.

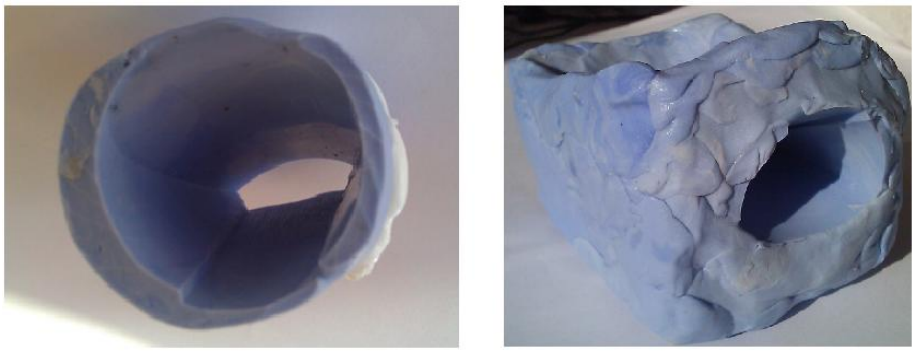


Figura 5-8: Primer i segon prototip de llavi.

Es va crear un primer model simple per comprovar que el sistema pogués funcionar (Figura 5-8). En primer pla aniria l'embocadura de l'instrument i a la part posterior un cilindre per enviar-hi la pressió de l'aire. Un segon prototip es crea amb una mena de capseta que permeti introduir-li els altres elements com serien els dos electroimants per controlar la llengua i la pressió dels llavis.

La idea és que mitjançant el compressor i la vàlvula de pressió es pugui controlar i fer sonar el clarinet. Per controlar la pressió d'aire a l'instrument es va estudiar la possibilitat de fer un sistema amb una resistència variable del tipus NV DS1804 del fabricant *MAXIM* per a la sortida de la vàlvula de la mateixa manera que també s'utilitzaria per controlar la pressió que un altre l'electroimant exerciria sobre la boca termoplàstica.

Aquest xip es tracta potenciòmetre digital no-volàtil que té 100 posicions. El dispositiu proporciona un mètode ideal per a aplicacions de baix cost amb una retallada de la CPU o el control manual amb un circuit extern mínim. La posició de cursor del DS1804 es pot emmagatzemar en la memòria EEPROM en la demanda.

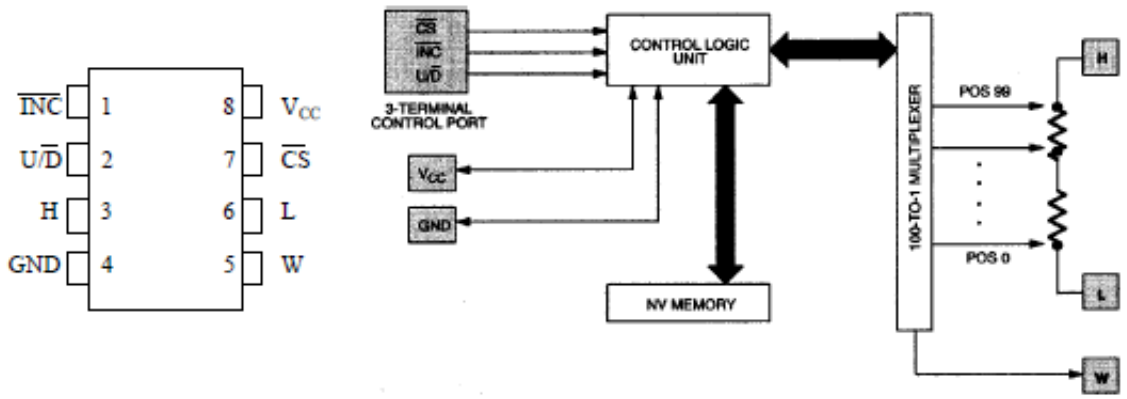


Figura 5-9: Assignació de pins i diagrama de blocs del DS1804.

El dispositiu compta amb un total de 100 posicions, un total de 99 segments de resistència incloent-hi les terminals L i H, que són accessibles per la terminal W.

El port de tres terminals de l'DS1804 ofereix una interfície d'increment/decrement que s'activa mitjançant una entrada de selecció de xip. Aquesta interfície es compon dels senyals d'entrada *CS, INC* i *U/D*. Aquests senyals d'entrada controlen un comptador de 7 bits. La sortida d'aquests 7 bits controlen un descodificador per seleccionar la posició del selector.

Aquesta resistència en serviria tant per controlar la pressió de la boca com per a controlar la vàlvula de pressió per deixar passar més o menys quantitat d'aire. Aquest xip es podrà controlar amb les sortides digitals del mateix *Arduino*, que en té 54, de les quals 19 ja estaran ocupades amb el sistema de teclat, 2 amb la boca i 3 amb el port de control de les tres terminals que regeixen la unitat lògica.

Pel que fa a l'aparell compressor, les proves s'han realitzat amb un model petit d'aerografia (Figura 5-10). Són dispositius no molt cars, no tenen un gran emmagatzematge d'aire però treuen una pressió suficient per fer sonar l'instrument.

Figura 5-10: Compressor d'aerografia amb el primer prototip de llavi (esquerra) i electrovàlvula (dreta).

`

L'últim element a explicar, doncs, serà l'electrovàlvula, que no és més que un electroimant que tallarà o no el pas de l'aire en un conducte. Es distingeixen dues classes: les normalment obertes, que quan està en repòs sense corrent deixen passar l'aire, i les normalment tancades, que deixen passar l'aire quan si li aplica corrent. En aquest cas s'ha utilitzat un normalment tancat ja que estarà més temps tancat que obert. S'ha escollit una electrovàlvula a 20V, igual voltatge que els electroimants.

Després d'haver seguit aquests passos i haver fet molts experiments amb diversos models de boques termoplàstiques i d'aconseguir algun bon resultat sonor, es va descartar continuar per aquest camí perquè no es va poder controlar els xiulets (galls) que es produeixen en moltes notes. A part que el sistema varia molt i es torna impracticable amb el més petit canvi de condicions, havent de fer retocs cada vegada que s'inicia una prova. A més, l'instrument escollit, el clarinet baix, no destaca per la seua facilitat d'execució.

5.3 Bloc de captació i processament

- Micròfons i Amplificadors

Com a sistema de captació és va fer proves amb un dels micròfon més corrent, el típic que s'utilitza per a sonoritzar instruments de vent en concerts. Es tracta del SM57 de la fàbrica Shure, un micròfon dinàmic, unidireccional i de baixa impedància.

Aquest micròfon té una sortida balancejada. Això significa que el cable del micròfon té dos conductors interns per portar el senyal minúscul del micròfon. Els dos cables estan envoltats per un conductor extern anomenat l'escut. L'escut no porta cap senyal. El seu treball consisteix a proporcionar un blindatge elèctric per mantenir la ràdio no desitjats i altres sorolls fora de l'interior de dos fils. Un cable balancejat donarà immunitat molt millor que el soroll elèctric i de ràdio que un cable de desequilibri.

La impedància nominal del micròfon és de 150 ohms (real de 310 ohms) per connectar amb de baixa impedància nominal -l'especificació de baixa impedància és d'entre 50 i 600 ohms-. El dispositiu escollit per amplificar la senyal d'aquest micròfon ha estat el xip INA217.

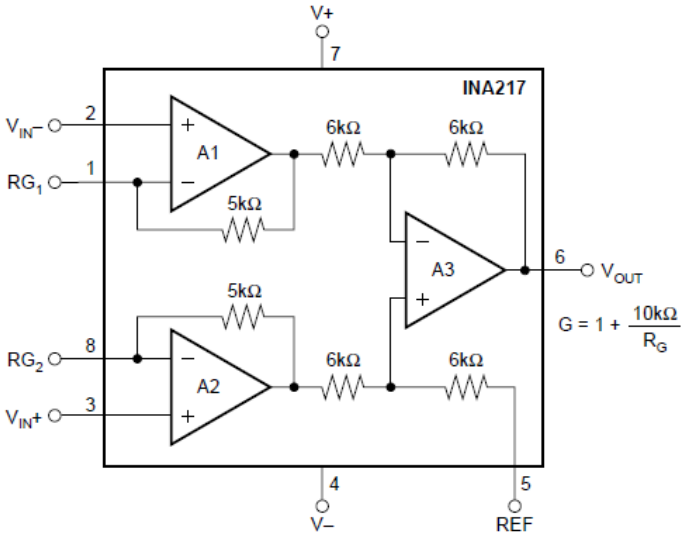


Figura 5-11: Esquema de xip INA217.

El INA217 és un amplificador d'instrumentació monolític de baix nivell de soroll i baixa distorsió. El corrent de retroalimentació permet que el circuit INA217 aconsegueixi un gran ample de banda i una excel·lent resposta dinàmica en un ampli rang de guany. És

ideal per a senyals d'àudio de baix nivell, el més equilibrat micròfons de baixa impedància.

El circuit únic cancel·lació de la distorsió redueix la distorsió a nivells extremadament baixos, fins i tot en els d'alt guany. El INA217 permet gairebé el funcionament teòric de soroll per a una micròfon d'impedància 200Ω.

Amb aquests elements el bloc de captació estaria totalment resolt. Tot i això també es van estudiar alternatives més econòmiques. No s'ha comprovat empíricament però segurament amb un simple electret unidireccional i un amplificador operacional seria suficient per l'entrada que necessitem ja que no seria necessari una qualitat tan bona de gravació perquè només interessa la freqüència fonamental.

Així doncs, amb l'ajuda d'un amplificador operacional es podria implementar també un circuit com el de la figura 5-12.

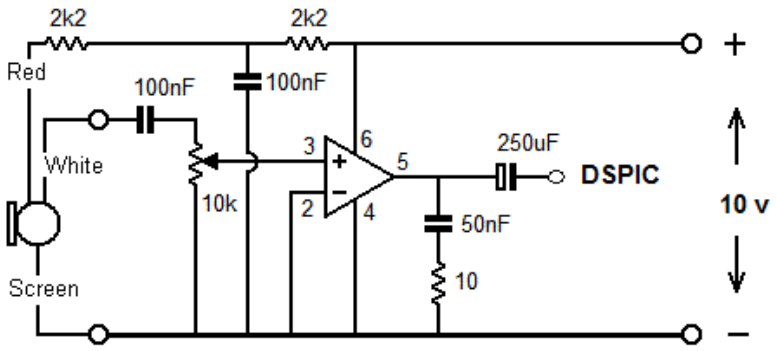


Figura 5-12: Esquema del preamplificador amb electret i amplificador operacional.

- **DSP**

Per altra banda, la idea inicial passava per construir una unitat independent amb plaques electròniques sense necessitat d'utilitzar un computador. Amb aquest objectiu, la segona línia de treball passaria per realitzar el processament mitjançant *DSP* - Processador de Senyals Digital- i realitzar la captació amb micròfons i amb amplificadors. Aquesta senyal amplificada serial directament la que vindria a l'entrada del DSP. En la fase inicial es van fer proves amb DSPic de la família 33 del fabricant Microchip.

Amb aquest dispositiu s'implementaria el sistema explicat a l'apartat 2.2. Un algoritme com el descrit en l'apartat 3 amb la transformada Q, no seria útil per treballar

a temps real a aquests dispositius. L'esquema general de blocs del DSPic es mostra en la següent figura 5-13.

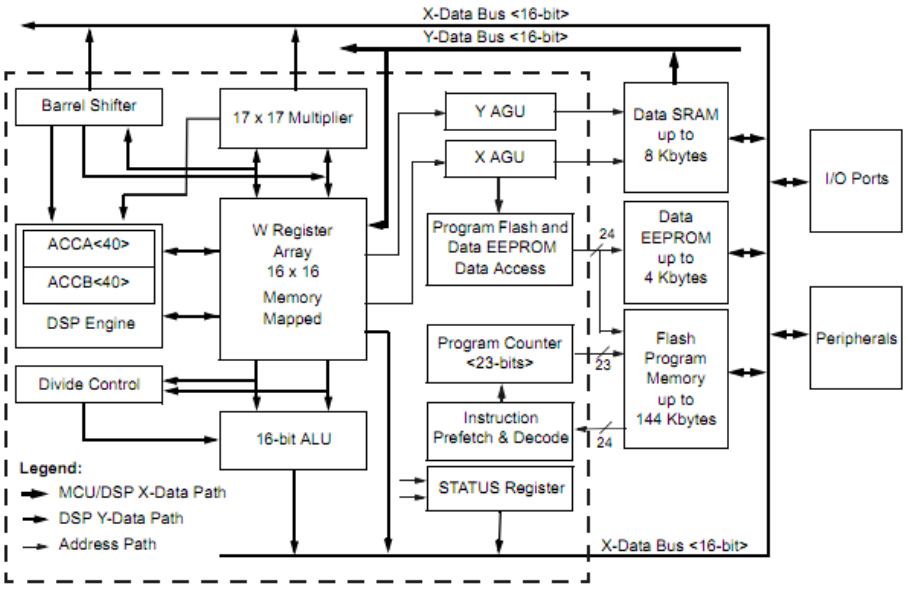


Figura 5-13: Diagrama de blocs de la família DSPic 30F.

Aquest DSPic té una CPU basada en una arquitectura de 16-bits de dades amb un gran nombre de instruccions que inclou suport per a realitzar algoritmes de processament digital de senyals. Aquesta CPU maneja paraules d'instrucció de 24-bits amb un camp de longitud variable d'acord al codi de la instrucció. El comptador de programa (PC) que té és d'una longitud de 24-bits i és direccionable fins 4M x 24 bits de l'espai de memòria de programa d'usuari. S'utilitza un mecanisme de pre-recerca, executable en un cicle d'instrucció, per ajudar a mantenir el correcte funcionament i proveir una execució predictible de la CPU.

Totes les instruccions són executables en un cicle d'instrucció, excepte les instruccions que canvien el flux del programa, les que mouen paraules dobles i les instruccions per a maneig de taules. Aquests dispositius tenen 16 registres de treball, cadascun de 16-bits, els quals poden ser registres de dades o adreces, addicionalment un 16è (*W register*, com és veu a la figura 5-13) s'ocupa com un punter de pila de programari per a les interrupcions i les crides.

El set d'instruccions dels DSPic30F compta amb dues classes d'instruccions, unes instruccions de funcionament MCU i altres de tipus DSP, les mateixes que s'inclouen en l'arquitectura i que s'executen amb una sola unitat d'execució. L'espai de dades pot adreçar paraules de 32K o bytes de 64K, les quals es divideixen en dos blocs coneguts

com a memòria de dades X i Y, els quals posseeixen la seva pròpia unitat generadora d'adreces (AGU).

Els 32Kbytes superiors de l'espai de la memòria de dades pot ser mapejat en l'espai de programa amb qualsevol de les 16K paraules de programa definides pels 8-bits del registre PSVPAG (Program Space Visibility Page).

Les maneres d'adreçament que suporta la CPU són inherents (sense operant), relatiu, literal, directe a memòria, directe a registres i adreçament indirecte a registres. Cada instrucció del DSPic es troba associada amb un o diversos modes d'adreçament predefinits d'acord amb les característiques de funcionament de la instrucció, ja que cada instrucció és capaç de treballar almenys amb sis modes d'adreçament.

Un xip d'aquest tipus és suficient per implementar l'algoritme descrit (Arvin, 2009). Com a sortida, la interfície perifèrica serial (SPI) permetrà enviar senyals MIDI directament. Ja que es tracta d'un mòdul d'interfície sèrie síncron per comunicar-se amb altres dispositius perifèrics o microcontroladors.

Una part no menys important serà la connexió entre la sortida del preamplificador i l'entrada del DSPic, aquesta part passaria per la conversió analògica digital. Aquest dispositiu pot tenir fins té 16 pins per transformar 16 senyals analògiques a digitals amb una resolució de 10 o 12 bits. Les entrades analògiques estan connectades a través de multiplexors cap l'amplificador S / H designat. El multiplexor d'entrada analògica es pot canviar entre dos conjunts d'entrades analògiques durant les conversions.

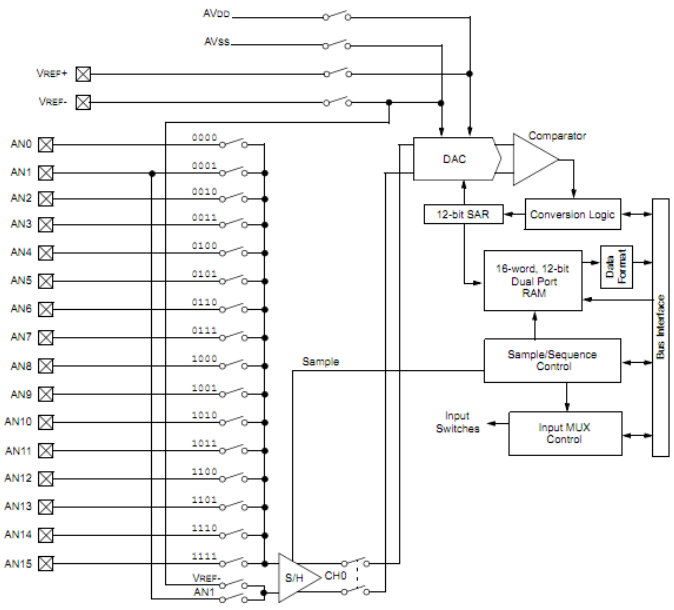


Figura 5-14: Conversos A/D de 12 bits.

Per desenvolupar aquesta idea, s'utilitzaria la conversió més acurada de 12 bits. Segons el manual el xip es necessita un temps d'adquisició de 14 períodes de rellotge, això comporta un mínim de 333,33 ns, com només s'utilitzaria una sola conversió, podríem arribar a tenir una freqüència de mostreig de 3 MHz. Aquest temps de conversió es podrà modificar amb determinats registres per tal de situar-lo al voltant de 44 KHz, que seria més que suficient.

Un altre paràmetre important seria aquest V_{ref} que apareix en la figura 5-15. Aquest pin serveix per indicar els valor d'entrada màxima i mínim de la senyal. Així es convenient implementar un circuit que detectés el màxim i el mínim per tal que es pogués ajustar automàticament la senyal d'entrada a uns cert llindar per tal d'aprofitar tot el rang de conversió.

Un circuit com el de la figura 5-13, serà útil a l'hora de situar aquests marges. Simplement canviant la polaritat a Vcc es pot obtenir el màxim i el mínim de l'entrada per situar el voltatge de referència per fer la conversió A/D.

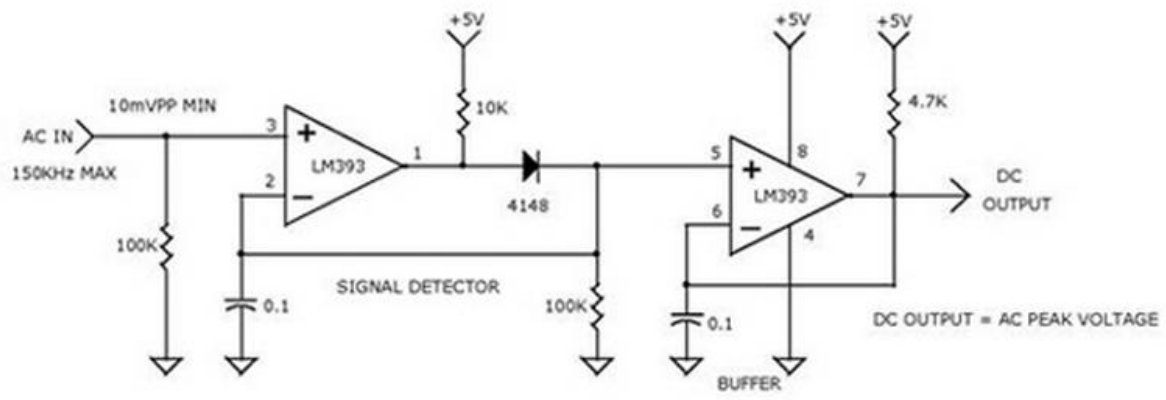


Figura 5-15: Detector de pic. [Font: <http://www.discovercircuits.com>]

6. Conclusions

Aquest projecte ha acabat amb la forma de programa per reproduir frases musicals en bucle, però no amb el mateix so, com els aparells *Loop-Station* que avui en dia hi ha al mercat, sinó buscant aconseguir la seqüència MIDI. Aquesta s'obté mitjançant el processament de la senyal captada i serà la que ens permetrà enviar la frase musical que s'enregistra a qualsevol dels múltiples instruments virtuals.

Tenint present els aparells Loop-Station que existeixen en el mercat s'ha creat una nova alternativa. Aparells com els de la figura 6-1, creats per la marca BOSS –filial de Roland S.A.- són els més populars dins el sector. Els tres models fan la funció fan d'enregistrar les seqüències d'àudio per després disparar-les.

Figura 6-1: Aparells RC-300 (Superior), RC-30 (Inferior esquerre) i RC-3 de la marca BOSS.

Aquests aparells es diferencien entre ells bàsicament amb el nombre de pistes estèreo independents que poden enregistrar: 1, 2 i 3 respectivament per als models RC-3, RC-30 i RC-300 respectivament. Cadascuna d'aquestes pistes pot estar composta per diverses seqüències sobreposades. Per exemple, es podria enregistrar primerament un ritme, després una línia de baix, després l'harmonia de la guitarra i amb tot això tenir la base per poder tocar al damunt. Aquesta característica és la principal funcionalitat que diferència aquests aparells d'allò implementat amb aquest projecte, que només permet processar un seqüència.

A més, el model RC-300 conté efectes de so, com poden ser *flanger, phaser, modulator*, efectes vocals, entre d'altres, amb un pedal d'intensitat per regular aquests paràmetres. Els tres models també tenen un bateria de ritmes que poden ser disparats. Aquestes funcionalitats no s'ha vist necessari implementar en aquest projecte ja que s'ha centrat amb la conversió àudio-MIDI i la caracterització de la senyal pel que fa bàsicament a *pitch*, vibrat, volum i duració.

Analitzant, doncs, aquesta conversió, el programa ha donat taxes de reconeixement del 90%, per a tempos no molt elevats. Tot i això s'ha comprovat que la no-detecció de notes no suposa que no es puguin reproduir, ja que l'algoritme implementat ha estat dissenyat per detectar els vibrats de nota. Amb això, es dóna el cas que algunes notes curtes s'analitzen conjuntament amb les seus notes contigües però que després el resultat sonor és satisfactori ja que s'envien senyals de canvi de *pitch*, que fan que aquestes notes apareguin.

Considerant la idea inicial del robot músic, s'ha optat, amb aquest programa Matlab, per realitzar una alternativa més coherent. Per a dur a terme el robot, es necessitaria un estudi molt més profund, no pel que fa al mecanisme digital, sinó a la part relativa al sistema artificial respiratori i, sobretot, a l'embocadura de l'instrument. Referent a la implementació del robot que controla el clarinet baix, els prototips inicialment creats no han estat capaços de donar resultats acceptables i, per tant, aconseguir un prototip que funcioni realment es deixa com a treball futur.

Una possible alternativa que es podria implementar seria la realització d'un prototip sobre un instrument més fàcil pel que fa a la execució i a la producció del so. El clarinet baix, tot i que dona un registre molt ampli i un color molt càlid, no destaca per la seua facilitat d'embocadura ni de so. Per aquest motiu, s'haurien hagut de considerar altres instruments del tipus EWI.

EWI (acrònim de *Electronic Wind Instrument*) és el nom del controlador de vent, un instrument musical electrònic inventat per Nyle Steiner. Els primers models consistien en dues parts: un controlador de vent i un sintetitzador. El model actual, EWI4000S, combina les dues parts en una sola, col·locant el sintetitzador a la part inferior del controlador. A més, utilitza el sistema de digitació Boehm i està dissenyat per a tenir una acció similar a un saxòfon soprano.

Cal remarcar que la part del controlador de vent de l'EWI té un filtre amb sensors de pressió d'aire (control de volum) i la pressió dels llavis (vibrato). Per altra banda, les tecles no es mouen, ja que treballen a través de la conductivitat, que detecta la posició dels dits per la corrent elèctrica, cosa que permet tocar molt ràpid.

Considerant aquest aparell, una implementació amb ell hagués estat molt més fàcil, ja que no hi ha cap element com és la canya del clarinet que s'ha de tractar d'una manera molt especial. Connectant directament l'electrovàlvula a l'entrada d'aquest instrument i amb un electroimant que controlés la pressió dels llavis estaria resolt. Tot i això aquest experiment no tindria cap gràcia musicalment ja que descodificaríem senyal MIDI per tornar a produir notes MIDI. Connectant la sortida del processament a aquest sintetitzador s'obtindria, en el millor dels casos, un resultat similar.

Pel que fa a línies de treball futures, la primera passaria per millorar el prototip, sobretot en quant al sistema que es posaria a l'embocadura de l'instrument, que ha estat el més difícil. Així doncs s'hauria de buscar la manera de com elaborar un prototip de boca més similar al desenvolupat per l'equip de la Universitat Waseda.

Una segona línia de treball futur seria passaria per la millora de l'algorisme de processament. Per tal d'assolir una taxa de reconeixement més elevada es podria implementar un sistema de finestra variable per a detectar de forma més robusta la freqüència fonamental, ja que quan s'augmenta la velocitat d'interpretació cada es van perdent més notes tal com es proposa en (Arvin, 2009). Per altra banda, l'adaptació de la funció correlació estreta per a alguns casos específics com les notes agudes -que comporten uns harmònics més propers un cop aplicada aquesta funció- podria també augmentar el grau d'encert.

Una tercera línia passaria per depurar la programació de la interfície de tal manera que quan es premés la icona de *Loop* per començar la reproducció no hagués d'esperar a què s'acabessin de processat les notes. Es tractaria de programar el codi utilitzant la concurrència per tal de què es poguessin enviar noves notes mentre està processant les últimes.

Com a última línia de treball es proposa millorar el codi de separació de fonts tal com s'explica a l'apartat (4.6), partint d'una base de dades construïda mitjançant diverses configuracions dels instrumentistes variant la distància entre ells i els micròfons. Això permetria que, quan el sistema està fent sonar el bucle, poder tocar i gravar una altra seqüència al damunt fent que el programa separi aquestes dues i pugui obtenir la nova seqüència neta per poder analitzar-la sense problemes. Els experiments realitzats en aquest projecte, tot i que baixar el nivell de la senyal contaminant, varien molt segons el tipus de distribució espacial.

7. Referències

- ARVIN, F.; DORAISAMY, S. *Australian Journal of Basic and Applied Sciences*. Real-Time Pitch Extraction of Acoustical Signals Using Windowing Approach, 2009, vol. 3, núm. 4.

- BEAUCHAMP, J. W. *Analysis, synthesis, and perception of musical sounds: the sound of music*. New York: Springer, 2007, p. 90-119.

- BITZER, Jörg. Mmidi + Source-Code [en línia]. Jade Hochschule, Institut für Hörtechnik+Audiologie. [Última consulta, 16 d'octubre de 2011]. Disponible a: <http://www.hoertechnik-audiologie.de/web/file/Forschung/Software.php>.

- BROWN, J.C.; PUCKETTE, M.S. *Journal of the Acoustical Society of America.* Calculation of a Narrowed Autocorrelation Function, 1989, vol. 85, p. 1595-1601.

- BROWN, J.C. *Journal of the Acoustical Society of America.* Calculation of a Constant Q Spectral Transform, 1991, vol. 89, núm. 1, p. 425-434.

- CICHOCKI, A.; AMARI, S.; *Adaptive Blind Signal and Image Processing: Learning Algorithms and Applications*. Chichester (England): John Wiley & Sons Ltd, 2002.

- HARMELING, S.; ZIEHE, A.; KAWANABE, M.; MÜLLER, K. *Neuronal Computation*. Kernel-Based Nonlinear Blind Source Separation, Maig 2003, Vol. 15, Núm. 5, Pàg. 1089-1124.

- HASS, Jeffrey. How midi works [en línia]. Indiana University, Jacobs School of Music, 2010. [Última consulta, 16 d'octubre de 2011]. Disponible a: <http://www.indiana.edu/~emusic/etext/MIDI/chapter3_MIDI.shtml>.

- KUO, SEN M.; LEE, BOB H. *Real-Time Digital Signal Processing: Implementations, Application and Experiments with the TMS320C55X.* Chichester: John Wiley & sons, 2001.

- PETERSEN, K.; SOLIS, J.; NINOMIYA, T.; YAMAMOTO, T.; TAKEUCHI, M; TAKANISHI, A. *IEEE International Conference on Robotics and Automation*. Development of the

Anthropomorphic Saxophonist Robot WAS-1: Mechanical Design of the Lip, Tonguing, Fingers and Air Pump Mechanisms, 2009, p. 3043-3048.

- QI, H.; HARTONOY, P.; SUZUKIZ, K.; HASHIMOTOX, S. *Acoustical Science and Technology*. Sound database retrieved by sound, 2002, vol. 23, núm. 6.

- SCHUMACHER, R. T. *Acustica.* Ab Initio Calculations of the Oscillations of a Clarinet, 1981, vol. 84, núm. 2, p. 71-85.

- SOLIS, J.; Petersen, K.; Ninomiya, T.; Takeuchi, M.; Takanishi, A. *IEEE/RSJ International Conference on Intelligent Robots and Systems.* Development of anthropomorphic musical performance robots: From understanding the nature of music performance to its application to entertainment robotics, 2009, p. 2309-2314.

- SOLIS, J.; TANIGUCHI, K.; NINOMIYA, T.; YAMAMOTO, T.; TAKANISHI, A. *Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. The Waseda Flutist Robot No. 4 Refined IV: Enhancing the sound clarity and the articulation between notes by improving the design of the lips and tonguing mechanisms, 2007, p. 2041 – 2046.

8. Annexos

8.1 Taula d'equivalències

La taula següent mostra les equivalències entre les notes, la freqüència absoluta i relativa, el nombre de nota midi corresponent i la posició teòrica dins del vector correlació. També amb color vermell i verd es mostra el registre del clarinet baix i el clarinet soprano respectivament.

$$Frel = \frac{Fabs}{Fm} = \frac{Fabs}{44100} \quad (9.1) \qquad Pcorr = \frac{1}{Frel} \quad (9.2)$$

$$Pitch = round\left(12 \cdot \log_2 \frac{f}{f_0}\right) + 81 \quad (9.3) \qquad f = 2^{\frac{Pitch-81}{12}} \cdot f_{af}/f_m \quad (9.3)$$

La freqüència f_0 correspon a la freqüència del La4, normalment 440Hz, que es podrà modificar (per exemple a vegades es pot afinar a 442Hz.). Val a dir que es pot considerar un factor d'octava (+-12) en les notes MIDI, teòricament hauria de el LA4 la nota MIDI 69, aquí s'ha considera a 81. Tot i que amb l'enviament de les notes MIDI es compensarà.

Nota	Fabs(Hz)	Frel	Pitch MIDI	Posició correlació	Cl. baix	Cl. sop.
A1	55.00	1.25E-03	45	801.82		
Bb1	58.27	1.32E-03	46	756.82		
B1	61.74	1.40E-03	47	714.34		
C2	65.41	1.48E-03	48	674.25		
Db2	69.30	1.57E-03	49	636.40		
D2	73.42	1.66E-03	50	600.68		
Eb2	77.78	1.76E-03	51	566.97		
E2	82.41	1.87E-03	52	535.15		
F2	87.31	1.98E-03	53	505.11		
F#2	92.50	2.10E-03	54	476.76		
G2	98.00	2.22E-03	55	450.01		
Ab2	103.83	2.35E-03	56	424.75		
A2	110.00	2.49E-03	57	400.91		
Bb2	116.54	2.64E-03	58	378.41		
B2	123.47	2.80E-03	59	357.17		
C3	130.81	2.97E-03	60	337.12		
Db3	138.59	3.14E-03	61	318.20		
D3	146.83	3.33E-03	62	300.34		
Eb3	155.56	3.53E-03	63	283.49		

Taula 8-1a: Equivalències de notes *pitch*.

E3	164.81	3.74E-03	64	267.57		
F3	174.61	3.96E-03	65	252.56		
F#3	185.00	4.19E-03	66	238.38		
G3	196.00	4.44E-03	67	225.00		
Ab3	207.65	4.71E-03	68	212.37		
A3	220.00	4.99E-03	69	200.45		
Bb3	233.08	5.29E-03	70	189.20		
B3	246.94	5.60E-03	71	178.58		
C4	261.63	5.93E-03	72	168.56		
Db4	277.18	6.29E-03	73	159.10		
D4	293.66	6.66E-03	74	150.17		
Eb4	311.13	7.06E-03	75	141.74		
E4	329.63	7.47E-03	76	133.79		
F4	349.23	7.92E-03	77	126.28		
F#4	369.99	8.39E-03	78	119.19		
G4	392.00	8.89E-03	79	112.50		
Ab4	415.30	9.42E-03	80	106.19		
A4	440.00	9.98E-03	81	100.23		
Bb4	466.16	1.06E-02	82	94.60		
B4	493.88	1.12E-02	83	89.29		
C5	523.25	1.19E-02	84	84.28		
Db5	554.37	1.26E-02	85	79.55		
D5	587.33	1.33E-02	86	75.09		
Eb5	622.25	1.41E-02	87	70.87		
E5	659.26	1.49E-02	88	66.89		
F5	698.46	1.58E-02	89	63.14		
F#5	739.99	1.68E-02	90	59.60		
G5	783.99	1.78E-02	91	56.25		
Ab5	830.61	1.88E-02	92	53.09		
A5	880.00	2.00E-02	93	50.11		
Bb5	932.33	2.11E-02	94	47.30		
B5	987.77	2.24E-02	95	44.65		
C6	1,046.50	2.37E-02	96	42.14		
Db6	1,108.73	2.51E-02	97	39.78		
D6	1,174.66	2.66E-02	98	37.54		
Eb6	1,244.51	2.82E-02	99	35.44		
E6	1,318.51	2.99E-02	100	33.45		
F6	1,396.91	3.17E-02	101	31.57		
F#6	1,479.98	3.36E-02	102	29.80		
G6	1,567.98	3.56E-02	103	28.13		
Ab6	1,661.22	3.77E-02	104	26.55		
A6	1,760.00	3.99E-02	105	25.06		
Bb7	1,864.66	4.23E-02	106	23.65		

Taula 8-2b: Equivalències de notes *pitch*.

8.2 Instruments i controladors MIDI

La taula següent mostra els corresponents instruments per a la instrucció MIDI canvi de programa de programa (http://www.midi.org/techspecs/gm1sound.php).

PC#	Nom d'Instrument	PC#	Nom d'Instrument	PC#	Nom d'Instrument	PC#	Nom d'Instrument
1.	Acoustic Grand Piano	65.	Soprano Sax	33.	Acoustic Bass	97.	FX 1 (rain)
2.	Bright Acoustic Piano	66.	Alto Sax	34.	Electric Bass (finger)	98.	FX 2 (soundtrack)
3.	Electric Grand Piano	67.	Tenor Sax	35.	Electric Bass (pick)	99.	FX 3 (crystal)
4.	Honky-tonk Piano	68.	Baritone Sax	36.	Fretless Bass	100.	FX 4 (atmosphere)
5.	Electric Piano 1	69.	Oboe	37.	Slap Bass 1	101.	FX 5 (brightness)
6.	Electric Piano 2	70.	English Horn	38.	Slap Bass 2	102.	FX 6 (goblins)
7.	Harpsichord	71.	Bassoon	39.	Synth Bass 1	103.	FX 7 (echoes)
8.	Clavi	72.	Clarinet	40.	Synth Bass 2	104.	FX 8 (sci-fi)
9.	Celesta	73.	Piccolo	41.	Violin	105.	Sitar
10.	Glockenspiel	74.	Flute	42.	Viola	106.	Banjo
11.	Music Box	75.	Recorder	43.	Cello	107.	Shamisen
12.	Vibraphone	76.	Pan Flute	44.	Contrabass	108.	Koto
13.	Marimba	77.	Blown Bottle	45.	Tremolo Strings	109.	Kalimba
14.	Xylophone	78.	Shakuhachi	46.	Pizzicato Strings	110.	Bag pipe
15.	Tubular Bells	79.	Whistle	47.	Orchestral Harp	111.	Fiddle
16.	Dulcimer	80.	Ocarina	48.	Timpani	112.	Shanai
17.	Drawbar Organ	81.	Lead 1 (square)	49.	String Ensemble 1	113.	Tinkle Bell
18.	Percussive Organ	82.	Lead 2 (sawtooth)	50.	String Ensemble 2	114.	Agogo
19.	Rock Organ	83.	Lead 3 (calliope)	51.	SynthStrings 1	115.	Steel Drums
20.	Church Organ	84.	Lead 4 (chiff)	52.	SynthStrings 2	116.	Woodblock
21.	Reed Organ	85.	Lead 5 (charang)	53.	Choir Aahs	117.	Taiko Drum
22.	Accordion	86.	Lead 6 (voice)	54.	Voice Oohs	118.	Melodic Tom
23.	Harmonica	87.	Lead 7 (fifths)	55.	Synth Voice	119.	Synth Drum
24.	Tango Accordion	88.	Lead 8 (bass + lead)	56.	Orchestra Hit	120.	Reverse Cymbal
25.	Acoustic Guitar (nylon)	89.	Pad 1 (new age)	57.	Trumpet	121.	Guitar Fret Noise
26.	Acoustic Guitar (steel)	90.	Pad 2 (warm)	58.	Trombone	122.	Breath Noise
27.	Electric Guitar (jazz)	91.	Pad 3 (polysynth)	59.	Tuba	123.	Seashore
28.	Electric Guitar (clean)	92.	Pad 4 (choir)	60.	Muted Trumpet	124.	Bird Tweet
29.	Electric Guitar (muted)	93.	Pad 5 (bowed)	61.	French Horn	125.	Telephone Ring
30.	Overdriven Guitar	94.	Pad 6 (metallic)	62.	Brass Section	126.	Helicopter
31.	Distortion Guitar	95.	Pad 7 (halo)	63.	SynthBrass 1	127.	Applause
32.	Guitar harmonics	96.	Pad 8 (sweep)	64.	SynthBrass 2		

Taula 8-3: Instruments definits per la norma GM1. [Nota: GM1 no defineix les característiques reals dels sons, els noms entre parèntesis després de cada un dels sintes, pads i efectes de so són pensats només com a guia).]

8.3 Exemples de canvis de *pitch* i detecció

En la taula 8-3 es pot veure el resultat que a la seqüència de notes amb el clarinet soprano. Aquesta taula és el resultat d'aplicar la correlació i veure els canvis que és produeixen dins de la seqüència. Així doncs, s'ha pogut estudiar l'efecte per elaborar l'algoritme de detecció.

A continuació es mostra el resultat després d'aplicar l'algoritme d'anàlisi dissenyat. El resultat obtingut es molt positiu, doncs excepte un error en la posició 74, 83 i 90-91 detecta tot els casos correctament. Tindríem una taxa de reconeixement de prop del 90%, quatre notes errònies de 37.

Posició Inicial	Posició final	Núm. Nota
21	56	62
56	95	63
95	133	64
133	171	65
171	209	66
209	246	67
246	284	68
284	321	69
321	359	70
359	395	71
395	434	72
434	470	73
470	501	74
511	547	74
547	584	75
584	620	76
620	656	77
656	691	78
691	727	79
727	761	80
761	795	81
795	831	82
831	861	83
869	871	83
871	905	84
905	939	85
939	975	86
975	1008	87
1008	1044	88
1044	1078	89
1078	1111	90
1112	1133	91
1144	1146	90
1146	1181	91
1181	1213	92
1213	1248	93
1248	1308	94
1308	1312	95
1312	1377	96
1377	1408	97
1408	1420	98

Taula 8-4: Detecció de les notes vibrades de la taula 8-5.

Pos. inicial	Pos. Final	Núm. Nota	Duració	Energia	Llindar soroll
1	6	0	5	0,00E+00	0
6	10	1	4	4,56E-03	1,96E-03
10	18	0	8	2,53E-03	2,06E-03
18	19	70	1	3,50E-03	1,48E-03
19	20	1	1	7,32E+00	1,48E-03
20	56	62	36	5,16E+01	1,48E-03
56	95	63	39	1,49E+01	1,48E-03
95	133	64	38	1,62E+01	1,48E-03
133	171	65	38	5,90E+00	1,48E-03
171	209	66	38	6,06E+00	1,48E-03
209	246	67	37	1,94E+01	1,48E-03
246	284	68	38	3,50E+01	1,48E-03
284	321	69	37	1,16E+01	1,48E-03
321	359	70	38	3,65E+00	1,48E-03
359	371	71	12	2,50E+01	1,48E-03
371	373	70	2	2,17E+01	1,48E-03
373	395	71	22	2,76E+01	1,48E-03
395	434	72	39	4,91E+00	1,48E-03
434	438	73	4	3,52E+01	1,48E-03
438	439	72	1	1,57E+01	1,48E-03
439	444	73	5	2,85E+01	1,48E-03
444	447	72	3	6,87E+00	1,48E-03
447	470	73	23	3,47E+01	1,48E-03
470	499	74	29	3,75E+01	1,48E-03
499	500	73	1	2,79E-02	1,48E-03
500	507	1	7	1,69E-02	1,48E-03
507	511	0	4	2,45E-03	1,55E-03
511	547	74	36	1,52E+01	1,49E-03
547	584	75	37	8,19E+00	1,49E-03
584	590	76	6	2,84E+00	1,49E-03
590	592	75	2	1,30E+01	1,49E-03
592	602	76	10	8,01E+00	1,49E-03
602	604	75	2	2,99E+00	1,49E-03
604	608	76	4	1,14E+01	1,49E-03
608	610	75	2	2,68E+00	1,49E-03
610	620	76	10	1,24E+01	1,49E-03
620	632	77	12	6,97E+00	1,49E-03
632	634	76	2	4,88E+00	1,49E-03
634	638	77	4	3,49E+01	1,49E-03
638	640	76	2	1,19E+01	1,49E-03
640	656	77	16	5,46E+01	1,49E-03
656	668	78	12	1,66E+01	1,49E-03
668	669	77	1	1,29E+01	1,49E-03
669	674	78	5	3,35E+01	1,49E-03
674	676	77	2	1,64E+01	1,49E-03
676	681	78	5	3,59E+01	1,49E-03
681	682	77	1	3,30E+01	1,49E-03
682	691	78	9	3,50E+01	1,49E-03
691	727	79	36	1,33E+01	1,49E-03
727	744	80	17	2,17E+01	1,49E-03
744	745	79	1	1,47E+01	1,49E-03
745	761	80	16	2,64E+01	1,49E-03
761	795	81	34	3,74E+01	1,49E-03
795	831	82	36	3,25E+01	1,49E-03
831	861	83	30	2,43E+01	1,49E-03
861	865	1	4	2,05E-02	1,49E-03
865	867	0	2	2,81E-03	1,57E-03
867	868	1	1	3,39E-03	1,63E-03
868	869	85	1	3,15E-02	1,63E-03
869	871	83	2	1,85E+01	1,63E-03
871	905	84	34	4,88E+01	1,63E-03
905	922	85	17	1,55E+01	1,63E-03
922	923	84	1	1,50E+01	1,63E-03
923	939	85	16	1,76E+01	1,63E-03
939	975	86	36	6,98E+01	1,63E-03
975	1008	87	33	4,75E+01	1,63E-03
1008	1044	88	36	3,09E+01	1,63E-03
1044	1054	89	10	1,24E+02	1,63E-03
1054	1055	88	1	9,31E+01	1,63E-03
1055	1071	89	16	1,49E+02	1,63E-03
1071	1072	88	1	8,72E+01	1,63E-03
1072	1078	89	6	1,43E+02	1,63E-03
1078	1088	90	10	6,39E+01	1,63E-03
1088	1089	89	1	3,87E+01	1,63E-03
1089	1093	90	4	4,67E+01	1,63E-03
1093	1095	89	2	1,63E+01	1,63E-03
1095	1111	90	16	6,13E+01	1,63E-03
1111	1112	79	1	5,45E+01	1,63E-03
1112	1116	91	4	4,79E+01	1,63E-03
1116	1118	90	2	3,54E+01	1,63E-03
1118	1126	91	8	6,13E+01	1,63E-03
1126	1128	90	2	5,12E+01	1,63E-03
1128	1133	91	5	6,43E+01	1,63E-03
1133	1141	1	8	1,70E-02	1,63E-03
1141	1144	0	3	2,61E-03	1,68E-03
1144	1146	90	2	6,55E+01	1,80E-03
1146	1151	91	5	4,30E+02	1,80E-03
1151	1153	90	2	8,48E+01	1,80E-03
1153	1156	91	3	1,04E+02	1,80E-03
1156	1157	90	1	6,06E+01	1,80E-03
1157	1162	91	5	1,19E+02	1,80E-03
1162	1163	90	1	5,08E+01	1,80E-03
1163	1171	91	8	1,09E+02	1,80E-03
1171	1172	90	1	1,59E+01	1,80E-03
1172	1181	91	9	3,17E+01	1,80E-03
1181	1185	92	4	1,69E+02	1,80E-03
1185	1187	91	2	6,11E+01	1,80E-03
1187	1191	92	4	2,09E+02	1,80E-03
1191	1193	91	2	1,03E+02	1,80E-03
1193	1196	92	3	2,01E+02	1,80E-03
1196	1199	91	3	5,05E+01	1,80E-03
1199	1202	92	3	1,66E+02	1,80E-03
1202	1204	91	2	8,08E+01	1,80E-03
1204	1213	92	9	2,22E+02	1,80E-03
1213	1218	93	5	1,72E+02	1,80E-03
1218	1220	92	2	1,17E+02	1,80E-03
1220	1223	93	3	2,05E+02	1,80E-03
1223	1226	92	3	1,32E+02	1,80E-03
1226	1229	93	3	1,80E+02	1,80E-03
1229	1232	92	3	1,38E+02	1,80E-03
1232	1235	93	3	1,91E+02	1,80E-03
1235	1237	92	2	8,77E+01	1,80E-03
1237	1241	93	4	1,77E+02	1,80E-03
1241	1243	92	2	1,84E+02	1,80E-03
1243	1248	93	5	2,18E+02	1,80E-03
1248	1252	94	4	1,49E+02	1,80E-03
1252	1253	93	1	1,36E+02	1,80E-03
1253	1257	94	4	1,54E+02	1,80E-03
1257	1260	93	3	1,11E+02	1,80E-03
1260	1262	94	2	6,79E+01	1,80E-03
1262	1265	93	3	9,60E+01	1,80E-03
1265	1268	94	3	1,03E+02	1,80E-03
1268	1271	93	3	8,17E+01	1,80E-03
1271	1274	94	3	9,40E+01	1,80E-03
1274	1275	93	1	1,25E+02	1,80E-03
1275	1281	94	6	1,51E+02	1,80E-03
1281	1284	95	3	2,28E+02	1,80E-03
1284	1287	94	3	8,81E+01	1,80E-03
1287	1289	95	2	1,27E+02	1,80E-03
1289	1291	94	2	1,73E+02	1,80E-03
1291	1293	95	2	2,11E+02	1,80E-03
1293	1297	94	4	9,44E+01	1,80E-03
1297	1299	95	2	1,25E+02	1,80E-03
1299	1302	94	3	6,62E+01	1,80E-03
1302	1305	95	3	1,81E+02	1,80E-03
1305	1308	94	3	8,19E+01	1,80E-03
1308	1312	95	4	1,46E+02	1,80E-03
1312	1317	96	5	1,93E+02	1,80E-03
1317	1318	95	1	1,67E+01	1,80E-03
1318	1330	96	12	5,47E+00	1,80E-03
1330	1332	95	2	1,15E+02	1,80E-03
1332	1335	96	3	1,79E+02	1,80E-03
1335	1337	95	2	8,38E+01	1,80E-03
1337	1350	96	13	2,10E+02	1,80E-03
1350	1351	97	1	2,02E+02	1,80E-03
1351	1355	96	4	1,33E+02	1,80E-03
1355	1357	97	2	1,52E+02	1,80E-03
1357	1361	96	4	8,45E+01	1,80E-03
1361	1362	97	1	1,18E+02	1,80E-03
1362	1366	96	4	5,16E+01	1,80E-03
1366	1368	97	2	2,27E+02	1,80E-03
1368	1372	96	4	1,34E+02	1,80E-03
1372	1373	97	1	1,60E+02	1,80E-03
1373	1377	96	4	1,11E+02	1,80E-03
1377	1381	97	4	2,47E+02	1,80E-03
1381	1384	98	3	2,19E+02	1,80E-03
1384	1387	97	3	2,02E+02	1,80E-03
1387	1389	98	2	9,41E+01	1,80E-03
1389	1392	97	3	9,77E+01	1,80E-03
1392	1394	98	2	1,04E+02	1,80E-03
1394	1398	97	4	4,69E+01	1,80E-03
1398	1400	98	2	7,35E+01	1,80E-03
1400	1404	97	4	1,00E+02	1,80E-03
1404	1406	98	2	7,03E+01	1,80E-03
1406	1408	97	2	1,09E+02	1,80E-03
1408	1417	98	9	9,62E+01	1,80E-03
1417	1418	86	1	2,24E-02	1,80E-03
1418	1420	98	2	8,87E-03	1,80E-03
1420	1421	102	1	3,79E-03	1,80E-03
1421	1421	0	0	1,16E-03	1,77E-03
1421	1421	0	0	1,16E-03	1,77E-03

Taula 8-5: Taula de canvis per al vibrat amb el clarinet soprà.

La següent taula és el resultat d'aplicar l'algoritme al clarinet baix. La taula ha estat tallada perquè es produïen massa canvis, es mostren els més importants.

Pos inicial	Pos final	Núm Nota	Du-ració	Energia	Llindar soroll
1	49	0	48	0,00E+00	0,00E+00
49	51	102	2	3,42E-03	1,22E-03
51	54	1	3	7,65E-03	1,22E-03
54	60	0	6	9,52E-04	1,22E-03
60	65	1	5	3,39E-03	1,16E-03
65	74	0	9	1,19E-03	1,16E-03
74	75	91	1	2,19E+01	1,08E-03
75	143	49	68	1,39E+02	1,08E-03
143	144	1	1	1,20E+01	1,08E-03
144	156	49	12	4,59E+01	1,08E-03
156	161	1	5	4,09E-01	1,08E-03
161	162	49	1	9,08E-03	1,08E-03
162	163	68	1	9,03E-03	1,08E-03
163	165	1	2	5,89E-03	1,08E-03
165	166	68	1	4,31E-03	1,08E-03
166	167	1	1	2,96E-03	1,08E-03
167	169	0	2	1,50E-03	1,08E-03
…	…	…	…	…	…
1050	1055	1	5	2,67E-03	1,21E-03
1055	1061	0	6	2,32E-03	1,22E-03
1061	1063	1	2	1,37E-02	1,21E-03
1063	1143	57	80	1,35E+01	1,21E-03
1143	1146	58	3	1,57E+01	1,21E-03
1146	1155	57	9	1,34E+01	1,21E-03
1155	1156	1	1	4,93E+00	1,21E-03
1156	1160	57	4	4,78E+00	1,21E-03
1160	1162	1	2	2,65E+00	1,21E-03
1162	1164	57	2	3,02E+00	1,21E-03
1164	1172	1	8	1,20E+00	1,21E-03
…	…	…	…	…	…
1335	1338	1	3	3,26E-03	1,17E-03
1338	1408	59	70	5,01E+01	1,17E-03
1408	1410	1	2	5,36E+00	1,17E-03
1410	1413	59	3	5,38E+00	1,17E-03
1413	1414	1	1	3,60E+00	1,17E-03
1414	1415	59	1	3,37E+00	1,17E-03
1415	1416	1	1	2,87E+00	1,17E-03
1416	1417	59	1	1,72E+00	1,17E-03
1417	1419	1	2	7,80E-01	1,17E-03
1419	1420	60	1	1,63E-01	1,17E-03
1420	1426	1	6	6,07E-02	1,17E-03
1426	1445	0	19	1,85E-03	1,18E-03
1445	1446	1	1	6,25E-01	1,16E-03
1446	1531	60	85	7,53E+01	1,16E-03
1531	1532	1	1	6,76E+00	1,16E-03
1532	1534	60	2	7,04E+00	1,16E-03
1534	1536	1	2	3,03E+00	1,16E-03
1536	1538	60	2	2,62E+00	1,16E-03
1538	1539	1	1	1,33E+00	1,16E-03
1539	1541	60	2	1,18E+00	1,16E-03
1541	1545	1	4	3,69E-01	1,16E-03
1545	1546	96	1	2,39E-03	1,16E-03
1546	1562	0	16	1,32E-03	1,16E-03
1562	1564	1	2	2,70E-03	1,14E-03
1564	1667	61	103	3,56E+01	1,14E-03
1667	1668	1	1	1,22E+00	1,14E-03
1668	1670	61	2	6,27E-01	1,14E-03
1670	1671	1	1	1,86E-01	1,14E-03
1671	1672	62	1	1,77E-02	1,14E-03
1672	1673	1	1	3,11E-03	1,14E-03
…	…	…	…	…	…
4590	4595	87	5	5,42E+02	1,22E-03
4595	4681	88	86	1,04E+03	1,22E-03
4681	4682	69	1	3,15E-02	1,22E-03
4682	4685	88	3	1,85E-02	1,22E-03
4685	4697	0	12	1,66E-03	1,22E-03
4697	4703	1	6	3,28E-03	1,21E-03

Taula 8-6: Taula de canvis per al vibrat amb el clarinet baix.

El resultat després d'aplicar el codi , és el que es mostra a la taula 8-7. Aquest és una mica millor que l'anterior doncs compta només amb 2 errors d'entre 42 notes.

Posició inicial	Posició final	Núm. Nota
75	156	49
178	259	50
282	373	51
388	493	52
512	620	53
642	747	54
776	885	55
916	1029	56
1064	1164	57
1208	1303	58
1339	1421	59
1447	1541	60
1565	1673	61
1696	1789	62
1807	1905	63
1922	2014	65
2022	2099	64
2111	2210	66
2221	2320	67
2328	2427	68
2439	2539	69
2555	2646	70
2673	2774	71
2788	2891	72
2911	3014	73
3032	3146	74
3158	3264	75
3277	3376	76
3396	3508	77
3526	3620	78
3637	3735	79
3747	3841	80
3853	3955	81
3962	4062	82
4073	4160	83
4170	4257	84
4270	4368	85
4377	4485	86
4485	4579	87
4591	4595	87
4595	4685	88
4711	4717	88
4717	4844	89
4851	4940	90

Taula 8-7: Detecció de les notes vibrades de la taula 9-4.

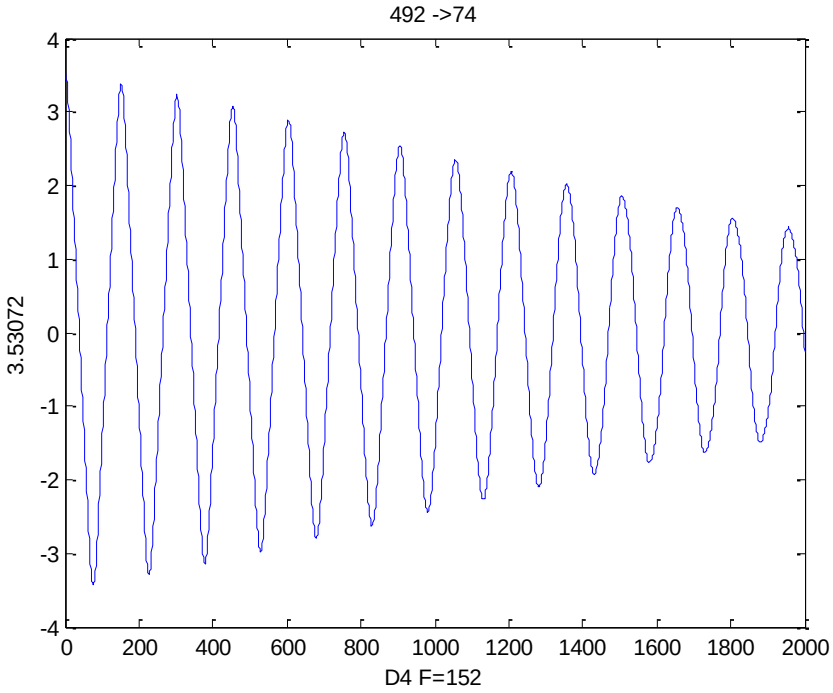
8.4 Codi de les gràfiques de correlació

Aquest codi obté les gràfiques de correlació per cada nota detectada.

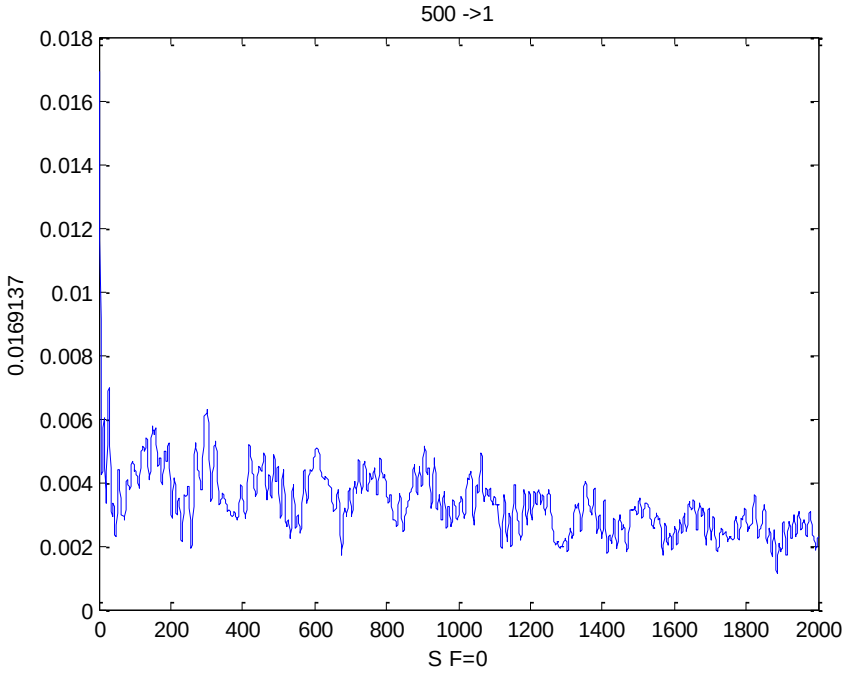
```
function prova(pos)
Y=ls('./wavs');     %carguem els arxius del directori wavs,
m=length(Y(:,1))-2; %m=numero d'arxius
windowLength=floor(100*44.1);   %tamany   de   la   finestra,
100ms,
step=floor(50*44.1); %tamany de cada pas, 50ms,

for i=1:m
    file=Y(i+2,:); %llegim el primer arxiu
    display(file)
    [x,Fs,bits] = wavread(strcat('wavs/',file));
    n=size(x);
    ch=n(2);
    n=n(1);
    if(ch==2)
        y=(x(:,1)+x(:,2));
    else
        y=x;
    end
    numOfFrames = floor((n-windowLength)/step) + 1;
    curPos=1+(pos-1)*step;
    for j=pos:numOfFrames
        window = (y(curPos:curPos+windowLength-1));
        [pitch,f,x]=corr_plot(window);
        nota=quina_nota(pitch);
        plot(x),xlabel(strcat(nota,'
F=',int2str(f))),ylabel(x(1)),title(strcat(int2str(j),'   ->
',int2str(pitch)));
        waitforbuttonpress;
        curPos = curPos + step;
    end
end
end
```

Figura 8-1: Codi per obtenir les gràfiques de correlació de les notes.

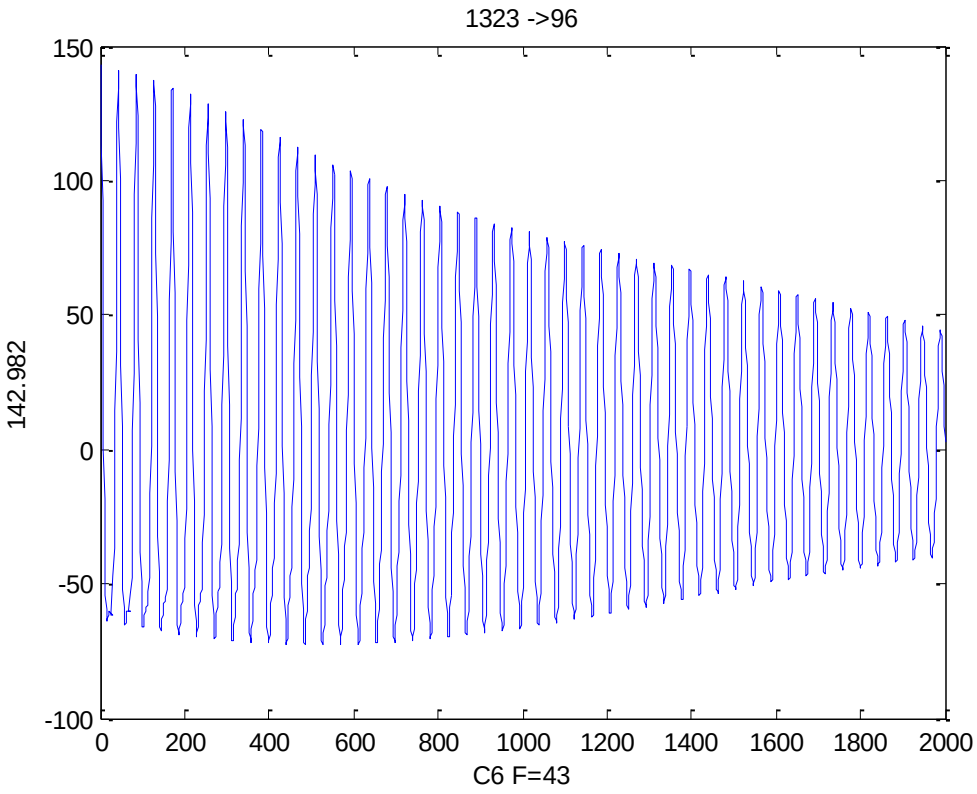
A continuació es presenten els resultats per a diferents notes captades quan s'interpretaven amb el clarinet soprano.



Gràfica 8-1: Correlació per a la nota Re4.



Gràfica 8-2: Correlació per al soroll.



Gràfica 8-3: Correlació per a la nota Do6.

Com es pot veure les gràfiques de correlació permeten una identificació molt fàcil i directa de nota i discriminen el soroll molt fàcilment amb els dos mètodes proposats en l'apartat 2.3.

8.5 Bloc central de processat i enviament

```
function main(handles)

%Declarem les variables locals que ens permetran interactuar amb els
% altres processos.
global mode notes AI y k llindar fTreball lLoop iMidi out tLoop;
global recPos curPos

trig=2500;  %variable que controla el nombre de mostres de captació
windowLength=floor(100*44);  %tamany de la finestra, 100ms,
step=floor(50*44);  %tamany de cada pas, 50ms,
curPos=1;  %variable de posició de lectura.
recPos=1;  %variable de posició d'escriptura.
t=1;    % controla la mitja del llindar del soroll
k=2;    % l'índex del vector notes
w=2;    % l'índex de les deteccions (notes + soroll)
y=[]; %vector que conté l'enregistrament del Loop
notes=zeros(100,3); %vector que contindrà les notes
    %notes(k,1)= posició inicial
    %notes(k,2)= posició final
    %notes(k,3)= pitch

lLoop=Inf; %Longitud del Loop, s'inicia a infinit.
iMidi=1;   %Index d'instrucció MIDI.
out=[];    %vector amb instruccions MIDI.
iNota=1;
Mnotes=[];
while curPos+windowLength<lLoop
    if mode==2
        if (recPos+trig)<(toc*44100)
                y(recPos:recPos+trig-1)=getdata(AI,trig);
                recPos=recPos+trig;
        end
        pause(0.01);
    elseif mode==3
        if recPos<lLoop
            y(recPos:lLoop-1)=getdata(AI,lLoop-recPos);
            stop(AI);
            delete(AI);
        end
        recPos=lLoop-1;
    end

    while curPos+windowLength<=recPos
        window = (y(curPos:curPos+windowLength-1));
        [pitch,E]=correlacio(window,llindar,fTreball);
        if pitch==0
            llindar=llindar*(t-1)/t+E/t;
            t=t+1;
        end
        if k>1 && pitch~=notes(k-1,3)
            notes(k,3)=pitch;
            notes(k,1)=curPos;
            if (curPos-notes(k-1,1))<17600
                if(notes(k-1,3)~=0)
                    if notes(k-2,3)==pitch
                        k=k-1;
```

```
                    elseif abs(notes(k-2,3)-notes(k-1,3))==1
                        notes(k-2,2)=curPos;
                        k=k-1;
                        notes(k,3)=pitch;
                        notes(k,1)=curPos;
                    elseif abs(notes(k-1,3)-pitch)<3 &&...
                            (curPos-notes(k-1,1))>step
                        notes(k-1,2)=curPos;
                        k=k+1;
                    else
                        notes(k-1,3)=pitch;
                        notes(k-1,1)=curPos;
                    end
                else
                    notes(k-1,2)=curPos;
                    k=k+1;
                end
            else
                notes(k-1,2)=curPos;
                k=k+1;
            end
        end
        curPos = curPos + step;
        if k-w>3
            if notes(w,3)>1
                queueMidi(notes(w,3),y,notes(w,1),notes(w,2),...
                    fTreball,64);
                Mnotes(iNota,:)=[notes(w,3),notes(w,1)/44100,...
                    notes(w,2)/44100];
                iNota=iNota+1;

            end
            w=w+1;
        end

    end
end
notes(k-1,2)=curPos;
for j=w:k-1
    if notes(j,3)>1
        queueMidi(notes(j,3),y,notes(j,1),notes(j,2),...
            fTreball,64);
        Mnotes(iNota,:)=[notes(j,3),notes(j,1)/44100,...
            notes(j,2)/44100];
        iNota=iNota+1;
    end
end
set(handles.TNotes,'Data',Mnotes);
guidata(handles.TNotes, handles);
toc
tic;
import javax.sound.midi.*
global MidiPort MidiInstrument r
info = MidiSystem.getMidiDeviceInfo; %Llista de dispositius
outputPort = MidiSystem.getMidiDevice(info(MidiPort)); %S'ha d'escollir un
dels de sortida de la llista anterior.
outputPort.open;
r = outputPort.getReceiver;
nMidi=1;
note = ShortMessage;
```

```
note.setMessage(ShortMessage.PROGRAM_CHANGE, 1, MidiInstrument, 1)
r.send(note, -1)

while mode==3
    if nMidi==1
        aux=out(nMidi).start;
    elseif nMidi<iMidi
        aux=out(nMidi).start-out(nMidi-1).start;
    else
        aux=tLoop-out(nMidi-1).start+out(1).start;
        nMidi=1;
        pause(0.001);
    end
    while toc<aux
    end
    tic;
    r.send(out(nMidi).note, -1)
    nMidi=nMidi+1;
end
outputPort.close;
```

8.6 Interfície gràfica (GUI Matlab)

```
function varargout = loop(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @loop_OpeningFcn, ...
                   'gui_OutputFcn',  @loop_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

function loop_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
global i t mode
i=0;
mode=0;
set(handles.Actual,'String',0);
set(handles.Panel,'BackgroundColor','red');
t = timer();
set(t, 'TimerFcn', {@TapCall, handles});


function varargout = loop_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

function Compas_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
global compas
compas=4;
set(hObject,'String',compas);

function Actual_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function Compas_Callback(hObject, eventdata, handles)
global compas
compas=str2double(get(hObject,'String'));

function ComUp_Callback(hObject, eventdata, handles)
global compas
compas=compas+1;
```

```
set(handles.Compas,'String',compas);

function ComDown_Callback(hObject, eventdata, handles)
global compas
compas=compas-1;
set(handles.Compas,'String',compas);


function Tap_Callback(hObject, eventdata, handles)
Metronom(handles)

function tapReset_Callback(hObject, eventdata, handles)
global i t
stop(t);
set(handles.Panel,'BackgroundColor','red');
i=0;

function notaAfina_Callback(hObject, eventdata, handles)

function notaAfina_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
set(hObject,'BackgroundColor','white');

function Afina_Callback(hObject, eventdata, handles)
Afinador(handles)

function freqAfina_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function freqAfina_Callback(hObject, eventdata, handles)

function fTreball_Callback(hObject, eventdata, handles)
global fTreball
fTreball=str2double(get(hObject,'String'));

function fTreball_CreateFcn(hObject, eventdata, handles)
global fTreball
fTreball=440;
set(hObject,'String',fTreball);
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function FUp_Callback(hObject, eventdata, handles)
global fTreball
fTreball=fTreball+0.5;
set(handles.fTreball,'String',fTreball);

function FDown_Callback(hObject, eventdata, handles)
global fTreball
```

```
fTreball=fTreball-0.5;
set(handles.fTreball,'String',fTreball);

function Loop_Callback(hObject, eventdata, handles)
global mode lLoop tLoop AI out iMidi nMidi

if mode==0
    tic;
    AI = analoginput('winsound');
    addchannel(AI,1);
    set(AI,'SampleRate',44100);
    set(AI,'SamplesPerTrigger',inf);
    start(AI);
    mode=2;
    set(hObject,'BackgroundColor','green');
    main(handles);

elseif mode==2
    tLoop=toc;
    lLoop=round(tLoop)*44100;
    mode=3;
    set(hObject,'BackgroundColor','red');
    tic;

elseif mode==3
    mode=0;
    set(hObject,'BackgroundColor','blue');
end


% --- Executes during object creation, after setting all properties.
function Loop_CreateFcn(hObject, eventdata, handles)
uicontrol(hObject,'Interruptible','on');


function TNotes_CreateFcn(hObject, eventdata, handles)
cnames = {'Nota','Inici','Final'};
set(hObject,'ColumnName',cnames)
set(hObject,'Data',[]);
guidata(hObject, handles);

function MidiPorts_Callback(hObject, eventdata, handles)
global MidiPort
MidiPort=get(hObject,'Value');

function MidiPorts_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
import javax.sound.midi.*
info = MidiSystem.getMidiDeviceInfo; %Llista de dispositius
aux=info(1).getName;

for i=2:length(info)
    aux=char(char(aux),char(info(i).getName));
end
set(hObject,'String',aux)
```

```
guidata(hObject, handles);


% --- Executes on selection change in istrMIDI.
function istrMIDI_Callback(hObject, eventdata, handles)
global MidiInstrument mode r
MidiInstrument=get(hObject,'Value')-1;
import javax.sound.midi.*
if mode==3
    note = ShortMessage;
    note.setMessage(ShortMessage.PROGRAM_CHANGE, 1, MidiInstrument, 1)
    r.send(note, -1)
end

% --- Executes during object creation, after setting all properties.
function istrMIDI_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
global MidiInstrument
MidiInstrument=1;
fid = fopen('midi.TXT');
tline = fgets(fid);
aux=tline;
while ischar(tline)
    tline = fgets(fid);
    aux=char(aux,tline);
end
fclose(fid);set(hObject,'String',aux)
guidata(hObject, handles);
```

8.7 Afinador

```
function Afinador(handles)
global mode MAX MIN llindar llindarQ fTreball
if mode==0
    t=1;
    llindar=1;
    MAX=0;
    MIN=10;
    mode=2;
    AI = analoginput('winsound');
    addchannel(AI,1);
    set(AI,'SampleRate',44100);
    set(AI,'SamplesPerTrigger',4410);
    set(handles.Afina,'BackgroundColor','green');
    start(AI);
    while mode==2
        V=zeros(1,256);
        y=getdata(AI);
        start(AI);
        [pitch,L] = correlacio(y,llindar);
        [E,p,f] = Q(y,pitch,440,256);
        if pitch>1
            if E>MAX
                MAX=E;
            elseif E<MIN
                MIN=E;
            end
        elseif pitch==0
            llindar=llindar*(t-1)/t+L/t;
            llindarQ=llindarQ*(t-1)/t+E/t;
            t=t+1;

        end
        set(handles.notaAfina,'String',(quina_nota(pitch)));
        axes(handles.axes)
        x = -127:128;
        if pitch>1
            V(p)=1;
            set(handles.freqAfina,'String',round(f*44100));
            if pitch==81
                fTreball=f*44100
            end
        else
            set(handles.freqAfina,'String',0);
        end
        plot(x,V);
        guidata(handles.Afina, handles);
    end
    stop(AI);
    delete(AI);
    clear AI;
elseif mode==2
    set(handles.Afina,'BackgroundColor','white');
    mode=0;
end
set(handles.fTreball,'String',fTreball);
end
```

8.8 Tempo

```
function Metronom(handles)
global compas i vTempo t mig
if i==0
    tic;
    compas=str2double(get(handles.Compas,'String'));
    i=1;
    vTempo(1,1)=0;
    mig=0;

elseif i<compas*2-1
    vTempo(1,i+1)=toc;
    vTempo(2,i)=vTempo(1,i+1)-vTempo(1,i);
    mig=mig*(i-1)/i+vTempo(2,i)/i;
    set(handles.Actual,'String',round(60/vTempo(2,i)));
    if i==compas*2-2
        set(t,'ExecutionMode','fixedRate','Period', mig);
    end
    i=i+1;
elseif i==compas*2-1
    start(t);
    set(handles.Actual,'String',round(60/mig));
    if mod(i,2)==1
        set(handles.Panel,'BackgroundColor','black');
    else
        set(handles.Panel,'BackgroundColor','blue');
    end
    i=i+1;
end

end
```